



MERCURY APPLICATION MAPPING™

Administration Guide

MERCURY™

Mercury Application Mapping

Administration Guide

Version 3.0

Mercury Application Mapping Administration Guide, Version 3.0

This manual, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Interactive Corporation, may be covered by one or more of the following patents: United States: 5,511,185; 5,657,438; 5,701,139; 5,870,559; 5,958,008; 5,974,572; 6,137,782; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332; 6,449,739; 6,470,383; 6,477,483; 6,549,944; 6,560,564; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912; 6,694,288; 6,738,813; 6,738,933; 6,754,701; 6,792,460 and 6,810,494. Australia: 763468 and 762554. Other patents pending. All rights reserved.

Mercury, Mercury Interactive, the Mercury logo, the Mercury Interactive logo, LoadRunner, WinRunner, SiteScope and TestDirector are trademarks of Mercury Interactive Corporation and may be registered in certain jurisdictions. The absence of a trademark from this list does not constitute a waiver of Mercury's intellectual property rights concerning that trademark.

All other company, brand and product names may be trademarks or registered trademarks of their respective holders. Mercury disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Mercury Interactive Corporation
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Toll Free: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (650) 603-5300

© 2004-2005 Mercury Interactive Corporation, All rights reserved

If you have any comments or suggestions regarding this document, please send them via e-mail to documentation@mercury.com.

Table of Contents

Welcome to Mercury Application Mapping Administration Guide ...	vii
Using this Guide.....	vii
Getting More Information	ix
Typographical Conventions.....	x
Chapter 1: User and Role Management	1
Defining a User Profile	2
Understanding User and Role Management	4
Users and Roles Workflow.....	5
Configuring Users.....	5
Configuring Roles.....	8
Assigning Access Rights for Users and Roles	10
Chapter 2: Event System Management	17
About Event System Management	18
The Event System Architecture	21
Event System Workflow	22
Configuring an Event Rule.....	23
Configuring a Rule Condition	26
Configuring an Action for a Rule.....	29
Event Rule Actions	30
Configuring a Time Rule	34
Defining Values for Additional Attributes	39
Saving Active Events Using Event Utilities	42
Raw and Active Event Attributes.....	45

Chapter 3: Discovery Management	51
About the Discovery Process	52
Understanding the Discovery Management Dialog Box	52
Configuring a Domain and a Discovery Scope	56
Configuring the Connection Data for a Protocol	59
Configuring a Module	63
Understanding the Pattern Editor	67
Editing a Pattern	69
Activating a Discovery Pattern	76
Protocol Definitions	77
Chapter 4: Discovery Process Configuration	89
About the Discovery Process Settings	89
Discovery Process Configuration Management	90
The Probe Gateway and Probe Manager Configuration File	91
Discovery Process Configuration Files	103
The Discovery System Directory Structure.....	110
Chapter 5: User Interface Settings Configuration	115
Configuring the User Interface Properties	116
Adding Customized Icons to Mercury Application Mapping.....	117
Chapter 6: Server Properties Configuration	121
Configuring the Server Subsystem Parameters	122
Starting and Shutting Down the Mercury Application Mapping Server	136
Mercury Application Mapping Log Files and Directories	137
Defining Log Properties.....	139
Reducing Log Load	141
Defining Server Login Data	141
Chapter 7: Topology Database Management	143
Building the Topology Database	143
Exporting the Topology Database	144
Exporting the Topology Database Automatically	145
Importing a Database File	146
Saving and Loading User-Defined Configurations	147
Adding Objects to the Topology Database.....	148

Chapter 8: Customized Package Creation	153
About Creating Customized Packages.....	154
What is a Package?	154
Package Structure.....	154
Package Location	157
Package Deployment.....	157
Dependencies Among Packages	157
Creating Customized Packages.....	160
Creating a Package.....	161
Verifying the Validity of a Customized Package.....	162
Uninstalling and Updating a Customized Package.....	164
Chapter 9: Task Scheduling.....	167
About Task Scheduling.....	167
Predefined Tasks	168
Creating a Task	169
Activating Tasks.....	170
Scheduling a Task.....	170
Scheduled Actions	175
Chapter 10: Dynamic Object States.....	177
Creating Dynamic Management States.....	177
Using States	179
Chapter 11: Oracle Configuration, Monitoring, and Tuning	
Guidelines	181
About Oracle Configuration, Monitoring, and Tuning Guidelines..	181
Oracle Configuration Guidelines	182
Monitoring and Tuning Guidelines.....	186
Chapter 12: Oracle Backup Guidelines	191
About Oracle Backup Guidelines	191
Backup Methods.....	192
Oracle Recovery Manager (RMAN).....	194
Customizing and Running the Cold Backup Script.....	195
Backup Scripts.....	197
Appendix A: Discovery Methods.....	207
Index.....	215

Table of Contents

Welcome to Mercury Application Mapping Administration Guide

This guide has been written to assist system administrators and integrators with the configuration of Mercury Application Mapping. It provides information regarding Mercury Application Mapping components and subsystems and includes basic and advanced configuration procedures.

Read this book if you are going to configure, implement, or maintain the Mercury Application Mapping system. As these functions are critical to the functioning of the system, they should be performed only by integrators or other personnel with working knowledge of Mercury Application Mapping.

Using this Guide

The guide contains the following parts:

Chapter 1 User and Role Management

Describes how to create and manage users and roles in Mercury Application Mapping.

Chapter 2 Event System Management

Provides an overview of the Mercury Application Mapping Event system and describes how to configure event and time rules and manage the Event system.

Chapter 3 Discovery Management

Describes how to run the discovery process and activate and edit discovery patterns.

Chapter 4 Discovery Process Configuration

Describes how to configure the parameters of the Probe Gateway, Probe Manager and discovery methods and provides information regarding the discovery system directory structure.

Chapter 5 Server Properties Configuration

Describes how to configure the server subsystem properties and how to define and use the system logs.

Chapter 6 User Interface Settings Configuration

Describes the configuration parameters and procedures needed to configure the Mercury Application Mapping interface.

Chapter 7 Topology Database Management

Describes the procedures and tools for creating and managing the Mercury Universal CMDB.

Chapter 8 Customized Package Creation

Describes how to install the Mercury Application Mapping default packages and how to create customized packages to suit your IT management needs.

Chapter 9 Task Scheduling

Describes how you can use Mercury Application Mapping's Scheduler to execute tasks to run on a periodic basis.

Chapter 10 Dynamic Object States

Describes how to dynamically define an object's management category.

Chapter 11 Oracle Configuration, Monitoring, and Tuning Guidelines

Describes the recommended Oracle database configurations and backup strategy for Mercury Application Mapping.

Chapter 12 Oracle Backup Guidelines

Describes the recommended Oracle backup strategy for Mercury Application Mapping.

Chapter 13 Discovery Methods

Lists the Mercury Application Mapping discovery methods that hold the logic for the discovery of IT infrastructure components from layers 2 through 7.

Getting More Information

For information on using the Mercury Application Mapping documentation set, reference information on additional documentation resources, and quick reference information on deploying, administering, and using Mercury Application Mapping, see *Getting Started with Mercury Application Mapping*.

Typographical Conventions

The guides that comprise the Mercury Application Mapping Documentation Library use the following typographical conventions:

- | | |
|-------------------|--|
| 1, 2, 3 | Bold numbers indicate steps in a procedure. |
| ► | Bullets indicate options and features. |
| > | The <i>greater than</i> sign separates menu levels (for example, File > Open). |
| Stone Sans | The Stone Sans font indicates names of interface elements in a procedure upon which you perform actions (for example: “Click the Run button.”), as well as directory and file paths. |
| <i>Italics</i> | <i>Italic</i> text indicates names (for example, names of variables or books). |
| Arial | The Arial font is used for examples and strings that are to be typed in literally. |
| <> | Angle brackets enclose a part of a URL address or file path that can vary (for example, <code>http://<hostname>/MercuryAM</code>). |
| [] | Square brackets enclose optional parameters. |

1

User and Role Management

This chapter describes how to define a user profile and create and manage users and user groups (called **roles** in Mercury Application Mapping).

This chapter describes:	On page:
Defining a User Profile	2
Understanding User and Role Management	4
Users and Roles Workflow	5
Configuring Users	5
Configuring Roles	8
Assigning Access Rights for Users and Roles	10

Defining a User Profile

Mercury Application Mapping enables the sharing of viewing settings among users, and allows you to personalize your own settings. By default, you can display only views you yourself created. However, each view and event filter that is created by one of the users in your system can be used by others. Thus, by using the User Profile dialog box you can use predefined views and filters and define your own user profile.

Note: Mercury Application Mapping also enables an Administrator to define profiles for other users in the User Manager dialog box. For details, see “Enabling an Administrator to Define a Profile for Other Users” on page 7.

To define your User Profile:

- 1** Select **Administration > User Management > User Profile** to open the User Profile dialog box with the **Views** tab displayed.

The User Profile dialog box saves and displays all the Views, Event Filters and Event Log Filters that are created by the system’s users.

- 2** Select the views and related events you want to display in your Map View:
 - ▶ Select **Show View** to display the selected view in the Map View.
 - ▶ Select **Get Event** to display the events related to the selected view in the Map View.
- 3** Click the **Event Filters** tab to select the event filters you want to use.
- 4** Click the **Event Log Filters** tab to select the filters you want to use.

Note: The filters you selected are available to you through the Filter Editor. For details, see “Using an Event Filter” on page 231.

5 Click the **Display** tab to set your display definitions.

- ▶ Select a **Label Transparency** option according to the following:
 - **Yes** – If you drag one object on top of another object so that the text of one label overlaps the other, the text of both labels remains visible. The default is Yes.
 - **No** – If you drag one object on top of another object so that the text of one label overlaps the text of the other label, the text of only one of the labels is clearly visible.
- ▶ In the **Max number of characters in a line** box, enter the maximum size (in characters) of item labels. In the following example, a maximum number of 15 characters is defined:



- ▶ In the **Max number of lines** box, enter the maximum number of lines of item labels. When the label is longer than the number of allocated characters, and is defined to spread over more than one line, the label text flows to the next line. In the following example, a maximum number of 10 characters and a maximum number of 3 lines is defined:



6 With the **Default Indicator Display option** buttons, you determine whether the available View Indicators (not including the dashboards) are to be displayed in 2-dimension or 3-dimension. For details on view indicators, see Chapter 15, “Generating View Indicators.”

Note: The **Window Time Range (min)** box is currently not in use.

- 7 Click **OK**. The views and related events you selected are displayed in the Explorer pane of your Map View.

Understanding User and Role Management

Access to Mercury Application Mapping functions is restricted to authorized users. You can assign specific permissions to one user or to a role. A role is a group of users to whom you assign the same access rights.

Mercury Application Mapping includes two default users and two default roles: Administrator and Guest. They can perform the following actions:

Administrator – Users and roles with administrator access rights have unrestricted access to all Mercury Application Mapping functions.

Guest – Users and roles with guest access rights have permissions to display views in the Map View.

When you create a user or role, you define what the user or role can perform in Mercury Application Mapping. You can assign those users or roles administrator or guest access rights, and if necessary, specify additional permissions in the Security Manager. For details, see “Assigning Access Rights for Users and Roles” on page 10.

For example, you can create a role that enables its users to create TQL queries, or you can create a role that enables its users to create views in the View Manager but not to edit existing queries.

Mercury Application Mapping enables you to edit the access rights of the default users and roles. If you upgrade your current Mercury Application Mapping version, the custom roles you have added remain in the system.

However, if you perform a new installation, your existing custom definitions are deleted. To save user or role ACL permissions for a new installation, use the packaging mechanism and save the information in the **acl.zip** file. For details on packaging, see “Chapter 8, “Customized Package Creation.” You must also recreate your user definitions.

Note: For security reasons, it is recommended to change the default definitions after the installation.

Users and Roles Workflow

You can create and manage users and assign roles to them (for details, see “Configuring Users” on page 5).

You can create and manage roles (for details, see “Configuring Roles” on page 8).

You can provide the user with access rights to each TQL query, managed view, domain instance, class model, and menu option for users and roles by assigning permissions to them (for details, see “Assigning Access Rights for Users and Roles” on page 10).

Configuring Users

This section explains how to create users and assign roles to them and how to manage them. A user takes on the role’s access rights. For details on roles, see “Configuring Roles” on page 8.

This section contains the following topics:

- ▶ “Creating a New User” on page 6
- ▶ “Enabling an Administrator to Define a Profile for Other Users” on page 7
- ▶ “Editing an Existing User” on page 7
- ▶ “Deleting an Existing User” on page 8

Creating a New User

To create a user, you define a name and password, and enter general user information such as telephone number, address, and so forth. Give each user a unique user name and password.

To create a new user:

- 1** Select **Administration > User Manager** to open the User Manager dialog box.
- 2** Click **Add** to open the Add New User dialog box.
 - ▶ In the **User Name** box, type the name of the user. Each user must have a unique name.
 - ▶ In the **Password** box, type the password of the user.
 - ▶ In the **Confirm Password** box, type the password again.
- 3** Click **Next** to enter the user's details.
- 4** In the subsequent dialog boxes, enter additional user details, if required.
- 5** Click **Next** to open the Role List dialog box.
- 6** In the **Role** list, select the role or roles you want to assign to the new user. The roles that appear in the **Roles list** also include any roles you created in the Roles dialog box (for details, see "Configuring Roles" on page 8).

If you do not select a role, you can specify permissions in the Security Manager (see "Assigning Access Rights for Users and Roles" on page 10).
- 7** Users with Administrator access rights can click the **Profile** button to define profiles for other users. For details, see "Enabling an Administrator to Define a Profile for Other Users" on page 7.
- 8** Click **Finish**. The new user is displayed in the User Manager dialog box.

Enabling an Administrator to Define a Profile for Other Users

Mercury Application Mapping allows users with administrator access rights to define profiles for other users, without having to log in and out of Mercury Application Mapping as that user. You can customize each user profile by choosing the view you want that user to see. When the user logs in to Mercury Application Mapping, only the views defined in the user profile are displayed. The administrator can change the user profile whenever required.

Note: If a user has conflicting access rights to a specific view in the Security Manager and User Profile, the settings in the User Profile override the settings in the Security Manager.

To define profiles for other users:

- 1 Select **Administration > User Manager** to open the User Manager dialog box.
- 2 Click the **Profile** button to open the User Profile dialog box. For details on how to define a user profile, see “Defining a User Profile” on page 2.

Editing an Existing User

You can edit an existing user’s properties.

To modify a user’s properties:

- 1 In the User Manager dialog box, select the user you want to edit and click **Edit** to open the Edit User Properties dialog box.
- 2 Repeat steps 3 to 8 in “Creating a New User” on page 6 to edit the information in the Personal, Business and Roles tabs.
- 3 Click **OK**.
- 4 In the User Manager dialog box, click **OK** again to save the changes you have made.

Deleting an Existing User

You can delete an existing user.

To delete a user:

- 1** In the User Manager dialog box, select the user you want to edit and click **Remove**.
- 2** Click **OK** to save the changes you have made.

Configuring Roles

Mercury Application Mapping enables you to create and manage roles. Each role in Mercury Application Mapping has permissions to perform specific actions according to the access rights assigned to it. You define the role name and description.

In addition to creating roles, you can edit and delete existing roles.

This section contains the following topics:

- ▶ “Creating a New Role” on page 8
- ▶ “Editing an Existing Role” on page 9
- ▶ “Deleting an Existing Role” on page 10

Creating a New Role

To create a role, you define a name and select existing permissions.

To create a new role

- 1** Select **Administration > User Management > Role Manager** to open the Roles dialog box.
- 2** Click **Add**.
- 3** In the **Name** box, type the name of the role.
- 4** If necessary, in the **Description** box, type a description of the new role.

- 5 To assign permissions, select the role(s) whose permission(s) you want to assign to the new role from the **Role List** or specify permissions in the Security Manager (for details, see “Assigning Access Rights for Users and Roles” on page 10).

Note: If you do not select a role from the **Role List**, you must assign permissions in the Security Manager to enable access rights for the role in Mercury Application Mapping. If you do select a role, any permissions you assign using the Security Manager in addition are added to the permissions already assigned to the new role.

- 6 Click **OK** to save the new role. The name of the role appears in the Roles dialog box.

Editing an Existing Role

You can edit an existing role’s capabilities.

To edit a role’s properties:

- 1 In the Roles dialog box, select the role you want to edit and click **Edit** to open the Role dialog box.
- 2 Make the necessary changes.
- 3 Click **OK** to save the changes you have made.

Deleting an Existing Role

This section describes how to delete an existing role.

Note: If you delete a role (Role A), whose permissions you assigned to another role (Role B), Role B will lose the permissions it was assigned from Role A.

To delete a role:

- 1 In the Role dialog box, select the role you want to delete and click **Remove**.
- 2 Click **OK** to save the changes you have made.

Assigning Access Rights for Users and Roles

You can assign access rights to each TQL query, managed view, domain instance, class model, and menu option for users and roles.

When assigning permissions to a user, you provide the user with access rights to the appropriate managers. For example, to give a user permissions for a view, give the user access rights to the View Manager. To give a user permissions to create TQLs, give the user access rights to the TQL Builder.

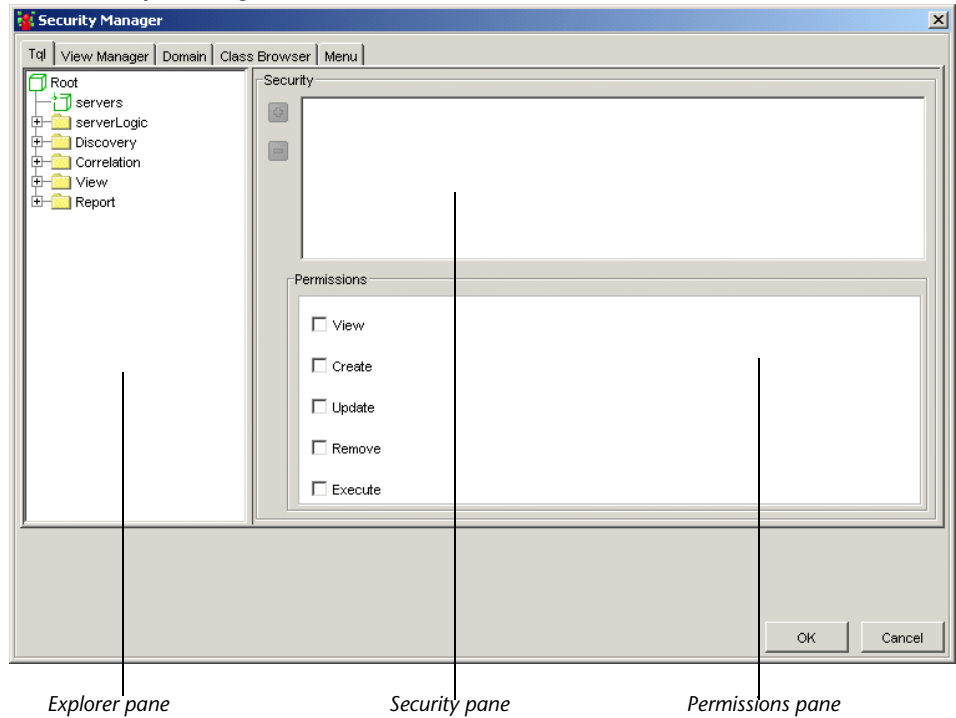
Note: A user that has created a new item in one of the Mercury Application Mapping managers, for example, a TQL query in the TQL Builder or a view in the View Manager, automatically has all permissions pertaining to that TQL query or view, such as view, update or remove.

This section contains the following topics:

- ▶ “Understanding the Security Manager” on page 11
- ▶ “Assigning Access Rights” on page 14
- ▶ “Removing Access Rights from Users and Roles” on page 15

Understanding the Security Manager

Select **Administration > User Management > Security Manager** to display the Security Manager window:



The following tabs display hierarchical tree structures:

- ▶ **Tql** – displays TQL queries
- ▶ **View Manager** – displays managed views
- ▶ **Domain** – displays domain instances
- ▶ **Class Browser** – displays the class model

► **Menu** – displays the menu items as follows:

Menu Name	Option	Description
Managers	Class Browser	Displays the managers to which you can assign access rights to users and roles. If a user or role does not have access rights to a certain manager, that manager does not appear in the Explorer pane during runtime.
	TQL Builder	
	View Manager	
	Report Manager	
	Correlation Manager	
	Logical Object Builder	
Map	Save	Save the changes you have made.
	Layer Setup	Define the view's layer layout.
	Change Password	Change the login password.
Edit	Filter Editor	Create and use existing event filters to reduce the number of events displayed in the Event Tabs, Browsers and Log.
	Find	Search for objects in the Map View, either in specific views and layers, or in the entire topology database.

Menu Name	Option	Description
Administration	Discovery Management	Display the Discovery Management dialog box, enabling you to activate selected discovery patterns and customize the discovery patterns and adapt them to your system's needs.
	Insert Object	Add objects to the database.
	User Manager	Create users.
	Role Manager	Create user groups.
	Security Manager	Define specific user or role permissions for Mercury Application Mapping resources.
	Show Concurrent Users	Display names of currently logged in users.
	Category Manager	Define an object's management category.
	Enumeration Manager	Create a predefined list of values.
	Event Configuration	Define the rules that govern the way changes in the state of managed objects and links are handled by the system.
	System Reports	Display the System Reports dialog box and select the report you want to view
	Scheduler	Define tasks to be activated on a periodic basis.
	Security Manager	Assign access rights for users and roles.

Assigning Access Rights

You can assign access rights to each managed view, domain instance, class model, and menu option for users and roles.

To assign access rights:

- 1 Select **Administration > User Management > Security Manager** to display the Security Manager window.
- 2 Select one of the following tabs and the item in the Explorer pane whose access rights you want to assign:
 - **Tql** – displays TQL queries
 - **View Manager** – displays managed views
 - **Domain** – displays domain instances
 - **Class Browser** – displays the class model
 - **Menu** – displays the menu items

Notes:

- You cannot select an item at the folder level.
 - If you make a selection at the root level, the user or role receives access rights to all items contained in the root.
-



- 3 The **Expand** button in the **Security** pane becomes available. Click the **Expand** button. A list of users and roles is displayed in the Roles dialog box.

The following table describes the icons and what they represent.

Icon	What it represents
A small blue icon of a single person.	A user
A small blue icon of two people.	A role

- 4 Select the users and/or the roles to which you want to assign access rights.

Note: You can use Windows conventions—**Shift+arrow** key, **Ctrl+arrow** key—to select adjacent or nonadjacent users.


- 5 Click **OK** to display the selected users and/or roles in the Security pane.
- 6 Select a user or role and select the check box in the Permissions pane next to each action that you want the user or role to perform.
- 7 Click **OK**. When users launch Mercury Application Mapping, only the actions for which you gave them permissions are available. For example, if a user is not assigned permissions to create a new role, the Role Manager option in the Administration menu is dimmed. If a user is not assigned permissions to view a certain manager, the manager does not appear in the Explorer pane.

Note: Child resources inherit the permissions assigned to their parents or the nearest ancestor to which permissions are assigned. For example, if you assign permissions to **host**, its children (**atwswitch**, **concentrator**, and so forth) inherit the permissions assigned to **host**.

Removing Access Rights from Users and Roles

You can remove permissions for user and roles.

To remove access rights:

- 1 Select **Administration > User Management > Security Manager:**
- 2 Select the relevant tab and the item in the Explorer pane whose access rights you want to remove.
- 3  Click the remove button.
- 4 Click **OK**.

2

Event System Management

This chapter describes the event system and provides definitions and instructions for its configuration and management.

This chapter describes:	On page:
The Event System Architecture	21
Event System Workflow	22
Configuring an Event Rule	23
Configuring a Rule Condition	26
Configuring an Action for a Rule	29
Event Rule Actions	30
Configuring a Time Rule	34
Defining Values for Additional Attributes	39
Saving Active Events Using Event Utilities	42
Raw and Active Event Attributes	45

About Event System Management

You can define rules to govern the way that Mercury Application Mapping handles changes in the state of managed objects and links. By defining these rules, you can determine the actions that the system should take when changes that meet predefined conditions occur, and how events reported to Mercury Application Mapping are translated into messages displayed in Mercury Application Mapping.

Raw Events and Active Events

The event system converts raw data about the state of the managed world into manageable information that can be viewed in Mercury Application Mapping. A raw event is a message that notifies Mercury Application Mapping about changes that are occurring in object and link attributes. The subsystems responsible for the notification are discovery, correlation, and time rules.

Raw events are saved in the Mercury Universal CMDB. If a raw event is related to an object that is displayed in the Map View, it appears in its event log.

Severity	Event Type	Class Name	Text Message	Create Time	Category
Plan	general_system_event	host	State was changed by Adminis...	06/24/2004 16:31:42	change
Major(7)	general_system_event	host	State was changed by Adminis...	06/24/2004 16:30:03	test
Critical	correlation_correlation.IP_Down.1...	host	ip is down	06/24/2004 16:27:48	operation

Unfiltered Ack: 0 Total: 3

Raw events that are related to one of the managed views are displayed in the Event Log

Note: For further details about the event log, see “Event Log” on page 228 in *Mercury Application Mapping User's Guide*.

From the Mercury Universal CMDB, the raw events are delivered, through JMS event messages, to the Event Configuration system. There, predefined event rules are applied to the raw events. The rules determine whether these raw events affect the system. Each of these event rules consists of at least one condition and at least one action. When a raw event meets one or more conditions, it triggers an action (or actions) that is performed by Mercury Application Mapping. The raw event then becomes an active event and is displayed in the Event browsers and tabs, and manifested via state changes and icons blinks in the Map View.

For example, an administrator defines a rule stating that only those events reported by the discovery system whose severity is **Critical** should be converted into active events in Mercury Application Mapping. In such a case, the action to be performed is **Create**, that is, an active event should be created. The administrator then defines another rule stating that when the severity on the object returns to **Normal**, the active event should be removed. In this case, the action to be performed is **Clear** or **Remove**.

Although the discovery, correlation, and time rules systems continuously create raw events in response to changes occurring in the managed world, only those that meet at least one of a rule's conditions are reported in Mercury Application Mapping.

Actions that are performed as a result of a condition fulfillment determine which raw events generate active events and which remain as raw events. One action, specifically aimed at creating active events from raw events, is the Create action, mentioned in the example above. Other actions that convert raw events into active events update the attributes of already existing active events. By performing this update, they simultaneously create new active events. These actions are called Internal Actions, and they include actions such as **Ack**, which update the acknowledgment related attributes of an active event, and **Suppress**, which updates the suppression related attributes.

You manage active events in the Map View where you can locate problems occurring at the object level. When an object contains an event (apart from events that are the result of a correlation rule) the icon for the selected object blinks and an exclamation mark is added before the icon in the Explorer pane. You can then click the **Object Events** tab or the other Event tabs and browsers to view the event information and acknowledge the event:

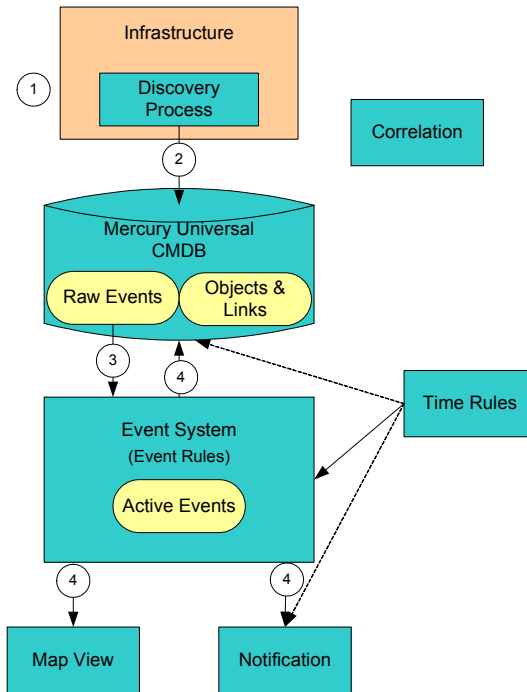
Event S...	Severity	Class Name	Object Global Label	Event Type	Text Message	Discovery Time	Acknowledged	Acknowledced
Critical	Critical	ip	10.0.64.20	general_syst...	State was change...	06/24/2004 16:27:47	<input checked="" type="checkbox"/>	Administrat
New	New	nt	ISRAEL-IBM	general_syst...	State was change...	06/24/2004 16:26:03	<input type="checkbox"/>	

Unfiltered | Ack: 1 | Total: 2

Once an event is acknowledged, the icon on the Map pane stops blinking and the exclamation mark disappears from the Explorer pane.

The Event System Architecture

The following figure illustrates the event system architecture.



The event flow is as follows:

- 1** A change occurs in the managed world. The discovery process identifies the involved object and collects data regarding the change.
- 2** Mercury Application Mapping creates a raw event, which contains the collected data and sends the event to the Mercury Universal CMDB where it is saved in the Raw Events repository.
- 3** The Mercury Universal CMDB reports the raw event to the event system.
If the change that occurred in the infrastructure is related to a correlation rule, the correlation system creates a raw event.

- 4** The Event System checks whether the raw event fulfils one or more event rule conditions. If the raw event fulfils a condition, the event system performs one or several of the following actions:
 - Updates the Mercury Universal CMDB with the raw event's data
 - Updates the Active Events repository, which can cause the creation of a new active event
 - Updates the views on the Map View
 - Sends a notification to a user or a group of users through the notification system regarding the raw event data
- 5** The Time Rules sub-system independently initiates a query on a periodic basis that searches the database for objects that meets its rules' conditions. If the change affects an object in a way that causes it to meet a time rule's condition, the time rule performs the actions described in step 4.

Event System Workflow

To configure the event system, you:

- Define event rules
For details, see “Configuring an Event Rule” on page 23.
- Define time rules
For details, see “Configuring a Time Rule” on page 34.
- Define shared and specific attributes
For details, see “Defining Values for Additional Attributes” on page 39.

Configuring an Event Rule

The Event Rules tab of the Event Configuration dialog box is used to define the conditions that must be met to trigger predefined actions to be performed by the system, such as creating an active event. When a raw event occurs somewhere in the managed world, Mercury Application Mapping examines the rules in this table, one by one, in order from top to bottom, until it finds one whose conditions are met by the raw event. It then executes the actions defined for that rule. Each rule can have multiple actions defined for it.

This section contains the following topics:

- ▶ “Understanding the Event Rules tab of the Event Configuration Dialog Box” on page 23
- ▶ “Defining an Event Rule” on page 24
- ▶ “Editing an Event Rule” on page 25
- ▶ “Deleting an Event Rule” on page 26

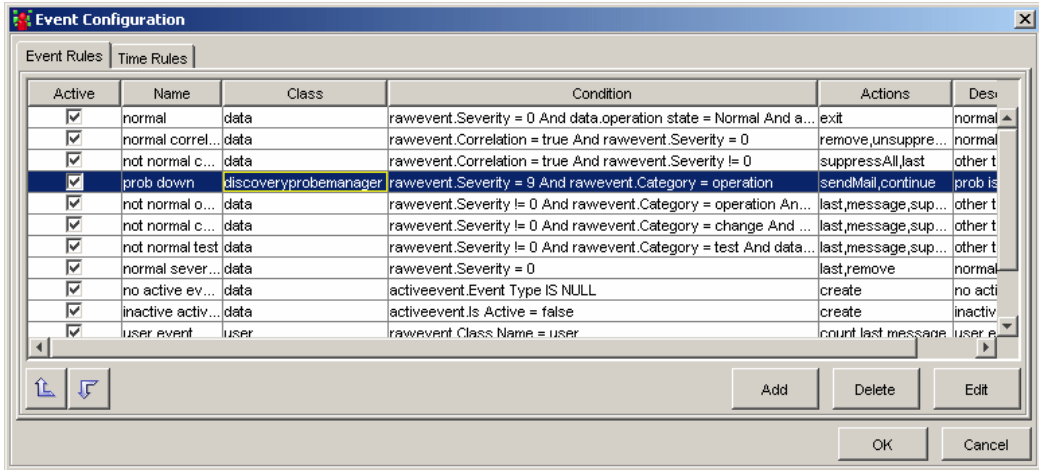
Understanding the Event Rules tab of the Event Configuration Dialog Box

Select **Administration > Event Configuration** to display the Event Configuration dialog box.

The Event Rules tab contains a table with the following columns:

- ▶ **Active** – Indicates which rules are currently active.
- ▶ **Name** – The unique name of the rule.
- ▶ **Class** – Display the class name of the related object.
- ▶ **Condition** – Displays the condition(s) of the rule.
- ▶ **Actions** – Displays the action(s) to be performed if the rule condition is met.
- ▶ **Description** – A description of the rule.

For example, in the following figure, the condition defined for the selected rule specifies that the occurrence of a raw event, whose class name is **discoveryproblemanager**, whose severity is **9** (critical), and whose category is **operation**, signifies that the Probe Manager is down.



Once these conditions are met, the following two actions are executed:



- ▶ **sendMail** – An e-mail is sent to the system administrator notifying that the Probe Manager is down.
- ▶ **continue** – Mercury Application Mapping continues to search the Event Rules list for a rule which acts as the trigger that changes the color of the object to red. Red reflects the severity of the event.

Defining an Event Rule

You can define an event rule.

To define an event rule:

- 1** Select **Administration > Event Configuration** to display the Event Rules tab of the Event Configuration dialog box.
- 2** Click **Add** to open the Event Rule dialog box.
- 3** Enter the name of the rule in the **Name** box and its description in the **Description** box in the **Rule Properties** area.

- 4 Select the object's class, to which the rule conditions are applied in the **Class** list. The list of available classes depends on the classes you or the user have defined.
 - 5 Specify the conditions of the rule in the **Conditions** area (for details, see "Configuring a Rule Condition" on page 26).
 - 6 Specify the actions of the rule in the **Actions** area of the Rule Action dialog box (for details, see "Configuring an Action for a Rule" on page 29).
 - 7 Click **OK** at the bottom of the Rule Action dialog box. The new rule is added to the table in the Event Rules tab of the Event Configuration dialog box. Verify that the **Active** check box of the new rule is selected. Only events generated by active rules are displayed in the Information pane of the Map View.
- 
- 
- 8 Use the up and down arrows to prioritize the rules in the Event Rules tab. Note that the order in which rules appear determines which rule is used for a specific event. The system performs the action(s) defined in the first relevant rule it finds in the list.
 - 9 Click **OK** to close the Event Configuration dialog box.

Tip: To restore the system default rules, click **Reset**.

Editing an Event Rule

You can edit an existing rule.

To edit an existing event rule:

- 1 Select **Administration > Event Configuration** to display the Event Rules tab of the Event Configuration dialog box.
- 2 Select the relevant rule in the Event Rules tab and click **Edit**.
- 3 Enter the changes (for details, see "Defining an Event Rule" on page 24)
- 4 Click **OK** to save the changes.

Deleting an Event Rule

You can delete an existing rule.

To delete an existing event rule:

- 1 Select **Administration > Event Configuration** to display the Event Rules tab of the Event Configuration dialog box.
- 2 Select the relevant rule in the Event Rules tab and click **Delete**.
- 3 Click **OK** to save the changes.

Configuring a Rule Condition

You can define the conditions that must be met to trigger predefined actions to be performed by the system, such as creating an active event.

This section contains the following topics:

- “Defining a Rule Condition” on page 26
- “Editing a Rule Condition” on page 28
- “Removing a Rule Condition” on page 28

Defining a Rule Condition

You can define a rule conditions.

To define a rule condition:

- 1 In the Event Rule dialog box, in the Conditions area, click **Add** to open the Event Rule dialog box.
- 2 At the top of the Condition dialog box, select the type of condition to define:
 - **object** – the attribute of a specific object
 - **rawevent** – the attribute of a specific raw event reported to Mercury Application Mapping
 - **activeevent** – the attribute of an active event already generated by Mercury Application Mapping

- 3** From the **Attribute** list, select the attribute to be included in the condition. For details, see “Raw and Active Event Attributes” on page 45.
- 4** From the **Operator** list, select the operator related to the attribute. Depending on the selected attribute, the following operators are available:

Operator	Description
=	Checks whether the attribute value is equal to the value specified in the value box.
!=	Checks whether the attribute value is not equal to the value specified in the value box.
>	Checks whether the attribute value is greater than the value specified in the value box.
>=	Checks whether the attribute value is greater than or equal to the value specified in the value box.
<	Checks whether the attribute value is less than the value specified in the value box.
<=	Checks whether the attribute value is less than or equal to the value specified in the value box.
LIKE	Uses a wildcard (%). Use Like when you are not sure of the complete name of what you are looking for.
NOT LIKE	Uses a wildcard (%). The same as ‘Like’ but looks for attribute values that do not include the string.
IN	Displays only the instances where this attribute value equals one of the selected values.
NOT IN	Displays only the instances where this attribute does not equal one of the selected values.
IS NULL	Checks whether the attribute value is null.
IS NOT NULL	Checks whether the attribute value is not null.
DURATION	The length of time during which the attribute value has not changed.

5 Select one of the following:

- ▶ **Value** – enter an absolute value to be included in the condition in the field provided
- ▶ **Parameter Value** – define a relative value by selecting a type (**object**, **rawevent**, or **activeevent**) and selecting the attribute related to that type from the **Parameter Attribute** list

6 Click **OK**. The condition is added to the table in the **Conditions** area of the Event Rule dialog box.

Tip: You can create complex rules by defining and combining several conditions for one rule.

7 If necessary, repeat this procedure to define additional conditions for the rule.

Editing a Rule Condition

You can edit existing conditions.

To edit existing rule conditions:

1 In the Event Rules dialog box, in the Conditions area, select the condition and click **Edit** to open the Condition dialog box.

For details, see “Configuring a Rule Condition” on page 26.

2 Click **OK** twice to return to the Event Configuration dialog box.

Removing a Rule Condition

You can remove existing conditions.

To remove existing rule conditions:

1 In the Event Rules dialog box, in the Conditions area, select the condition and click **Delete**.

For details, see “Configuring a Rule Condition” on page 26.

2 Click **OK** to return to the Event Configuration dialog box.

Configuring an Action for a Rule

Once Mercury Application Mapping has found the rules whose conditions are met by the raw event, it executes the actions defined for that rule. Each rule can have multiple actions defined for it. For details, see “Configuring an Event Rule” on page 23.

This section contains the following topics:

- “Defining a New Action for a Rule” on page 29
- “Removing an Action” on page 30

Defining a New Action for a Rule

You can define a new action for a rule.

To define a new action for a rule:

- 1** In the Event Rule dialog box, in the **Actions** area, click **Add** to open the Add New Action dialog box.
- 2** Select an action from the **Available Actions** list and click **Finish** (if no other configuration is required) or click **Next** (to define the specific parameters of the action to be performed in the boxes provided). For a description of available actions, see “Event Rule Actions” on page 30.

For example, if **Send Mail** is selected in the Available Actions list, you click **Next** to open a series of boxes that define the recipient, title, and message of the e-mail to be sent.
- 3** When done, click **Finish**. The action is added to the table in the Actions area of the Event Rule dialog box.
- 4** If necessary, repeat this procedure to define additional actions for the rule.

Tip: Use the up and down arrows in the **Actions** area to order multiple actions.

Removing an Action

You can remove an existing action.

To remove an existing action:

- 1** In the Event Rule dialog box, in the **Actions** area, select the action and click **Delete** to remove the action.
- 2** Click **OK** to save the change.

Event Rule Actions

Three categories of actions can be defined for a rule: admin actions, internal actions, and external actions.

- ▶ **Admin Actions** – the admin action category refers to administrative actions such as skipping an event.
- ▶ **Internal Actions** – the internal actions category refers to actions that affect either the active event, all the events related to a particular object, or the object itself. There are two types of internal actions: for active events or for objects. There are two types of internal actions:
 - ▶ **Internal Actions for Active Events** – internal actions for active events refers to actions that affect the active event.
 - ▶ **Internal Actions for Objects** – the internal actions for objects refers to actions that affect all the events related to a particular object or the object itself.
- ▶ **External Actions** – external actions refers to actions that take place outside the Mercury Application Mapping application.

Note: The actions defined for events and time rules do not generate raw events when activated. For example, if the action **Remove** has been performed, no raw event is generated or sent to the event log file. By contrast, user actions continue to generate raw events. For example, a user deleting an active event creates a raw event with the same parameters (ID, message, severity, and so forth).

Action	Description	Type of Action
ack	Updates the acknowledged , acknowledgment time , and acknowledged by attributes of the active event.	Internal actions for active events
ackAll	Updates the acknowledged , acknowledgment time , and acknowledged by attributes of all active events of the object	Internal actions for active events
addDiscovery Pattern	Displays the Add Discovery Pattern dialog box, enabling you to manually invoke a discovery pattern for the selected object. You can use this option to discover additional information about the object through one of the available discovery patterns. You can also use the Add Discovery Pattern dialog box to edit the parameters of the discovery patterns and to customize them to your needs.	Internal actions for objects
clear	Deactivates the active event by changing its active attribute to false. As a result, the active event is removed from the Event Browsers. However, the event can still be restored, for example by the Create and Escalate actions.	Internal actions for active events

Action	Description	Type of Action
clearAll	Updates the active and activation time attributes for all active events in the object	Internal actions for active events
continue	Continues with the next event rule in the table. This action always appears together with another action, and it instructs the system to continue checking the list for another matching condition.	Admin actions
count	Increments the event count attribute of the active event	Internal actions for active events
create	Creates or revives an active event	Internal actions for active events
escalate	Escalates the severity attribute of the active event, up to critical level	Internal actions for active events
execute	Runs an executable file, including defined parameters	External actions
exit	Does not continue checking the rule list for this event	Admin actions
last	Updates the system update time, discovery update time, severity, and text message attributes of the active event	Internal actions for active events
message	Updates the text message attribute of the active event	Internal actions for active events
remove	Deletes the active event	Internal actions for active events
removeAll	Deletes the object's active events	Internal actions for active events
remove Discovery Pattern	Displays the Remove Discovery Pattern dialog box, enabling you to manually remove a discovery pattern from the selected object	Internal actions for objects

Action	Description	Type of Action
removeObject	Removes the object and all its active events	Internal actions for objects
replace	Updates the severity attribute of the active event and increments the event count attribute	Internal actions for active events
reset	Resets the event count attribute of the active event	Internal actions for active events
sendEvent	Sends an event. The user can define the severity and content of the message.	External actions
sendMail	Sends an e-mail. The user can define the recipient, title and message of the e-mail.	External actions
setActiveEven Attribute	Sets the value of one of the active event's attributes	Internal actions for objects
setAttribute	Sets the attribute of the object, where the attribute must be in a class other than root or data	Internal actions for objects
suppress	Updates the suppressed and suppression time attributes of the active event	Internal actions for active events
suppressAll	Updates the suppressed and suppression time attributes for all active events in the object	Internal actions for active events
time	Updates the system time and discovery time attributes of the active event	Internal actions for active events
unack	Updates the acknowledged , acknowledgment time , and acknowledged by attributes of the active event	Internal actions for active events
unackAll	Updates the acknowledged , acknowledgment time , and acknowledged by attributes for all active events in the object	Internal actions for active events

Action	Description	Type of Action
unsuppress	Updates the suppressed and suppression time attributes of the active event	Internal actions for active events
unsuppressAll	Updates the suppressed and suppression time attributes for all active events in the object	Internal actions for active events

Configuring a Time Rule

The Time Rules tab of the Event Configuration dialog box is used to define event rules that act on a periodic basis and query the Mercury Universal CMDB about the state of stored objects and events. You define time rules that act on a periodic basis and query the Mercury Universal CMDB about the state of stored objects and events by specifying at least one condition, at least one action, and an activation interval.

This section contains the following topics:

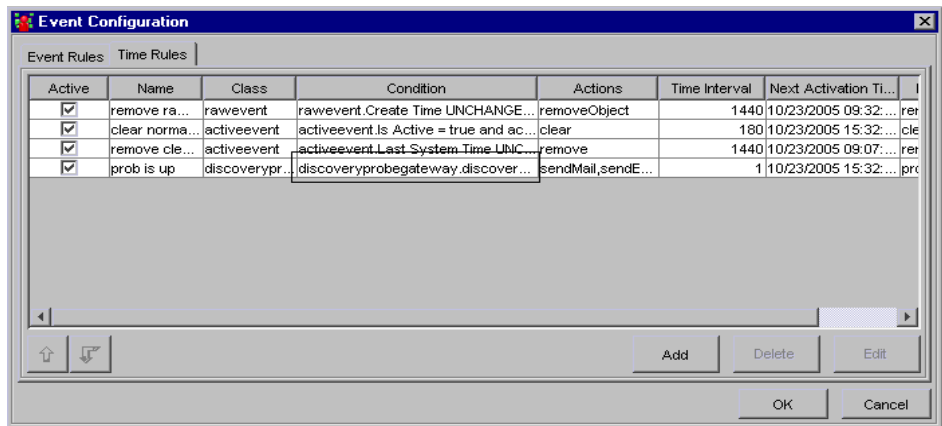
- “Understanding the Time Rules tab of the Event Configuration Dialog Box” on page 35
- “Defining a Time Rule” on page 36
- “Editing a Time Rule” on page 38
- “Deleting a Time Rule” on page 38

Understanding the Time Rules tab of the Event Configuration Dialog Box

An event rule that is defined in the Event Rules tab is checked only when a new raw event arrives. The arrival of a raw event is caused by a state change in one of the managed objects; when the change meets the rule's condition, the rule's action is applied to the object.

Unlike this type of event rule, a time-based event rule that is created in the Time Rules tab is checked regularly, according to a predefined time interval. Each time the rule is checked, it initiates a query that searches the database for objects that meet the rule's condition. When the condition is fulfilled, an action is triggered that is applied to the managed objects whose state meets the rule's condition.

For example, a time-based event rule states that if the Probe Gateway is down for more than ten minutes, an e-mail should be sent to inform the administrator about the probe's state. The following figure illustrates such a rule. The selected rule in the **Time Rules** list specifies that if the probe's (proxy) **discoveryprobegateway** attribute does not change in the course of ten minutes, the rule should trigger an action that sends an e-mail regarding this state:



This rule also contains a time interval for its activation – every ten minutes. This means that every ten minutes a query is built and sent to the database to search for changes that might have occurred in the **discoveryprobegateway** attribute. The date and hour that the rule should be next activated are specified in the Next Activation Time column.

Another time based event rule is added to the above event system. This rule states that if the status of an object has been critical for an extended period of time (30 days), the object must be removed from the system.

The Time Rules table contains the following columns:

- ▶ **Active** – for defining which rules are currently active
- ▶ **Name** - the unique name of the rule
- ▶ **Class** – displays the class name of the related object, to which the condition is applied
- ▶ **Condition** – displays the condition(s) of the rule
- ▶ **Actions** – displays the action(s) to be performed if the rule condition is met
- ▶ **Time Interval** – for defining the time interval that passes between each examination of the rule
- ▶ **Next Activation Time** – displays the next date and time of the rule activation
- ▶ **Description** – the description of the time rule

Defining a Time Rule

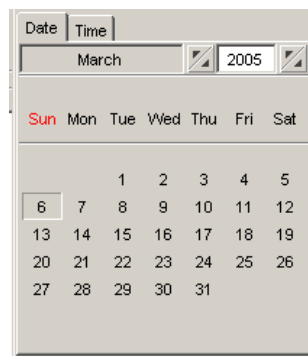
You can define a time rule.

To define a time rule:

- 1** Select **Administration > Event Configuration** to open the Event Configuration dialog box.
- 2** Select the **Time Rules** tab.
- 3** Click **Add** to open the Time Rule dialog box.

4 In the **Rules Properties** area:

- ▶ Enter the name of the rule in the **Name** box and its description in the **Description** box in the **Rule Properties** area.
- ▶ In the **Class** list, select the class of the object to which the rule conditions should be applied. The list of available classes depends on the classes you or the user have defined.
- ▶ In the **Conditions** area, click **Add** to open the Condition dialog box (for details, see “Configuring a Rule Condition” on page 26).
- ▶ In the **Time Interval** area, proceed as follows:
 - In the **Time Interval** box, enter the time interval (in minutes) that passes between each examination of the rule.
 - The time and date in the **Next Activation Time** box are automatically updated according to your Time Interval definition and your current date and time. However, you can manually specify a different date and time for the next activation of the rule by clicking on the window to open a calendar.



- 5** In the **Actions** area of the Rule Action dialog box, click **Add** to open the Add New Action dialog box (for details, see “Configuring an Action for a Rule” on page 29).
- 6** When you have finished defining the conditions, time interval and actions for the time-based event rule, click **OK** in the Time Rule dialog box. The rule you defined is displayed in the Time Rules tab of the Event Configuration dialog box.

Note: The up and down arrows are disabled in the Time Rules tab as time-based event rules do not need to be prioritized: the defined time interval acts as a type of prioritization.

Editing a Time Rule

You can edit an existing time rule.

To edit an existing time rule:

- 1** Select **Administration > Event Configuration** to open the Event Configuration dialog box is displayed (for details, see “Configuring an Event Rule” on page 23).
- 2** Click the **Time Rules** tab, select the time rule, and click **Edit** (for details, see “Configuring an Event Rule” on page 23).

Deleting a Time Rule

You can delete an existing time rule.

To delete an existing Time Rule:

- 1** Select **Administration > Event Configuration** to open the Event Configuration dialog box is displayed, as shown in “Configuring an Event Rule” on page 23.
- 2** Click the **Time Rules** tab, select the time rule, and click **Delete**.

Defining Values for Additional Attributes

Active events are always created from raw events (for details, see “Raw Event Attributes” on page 46) and both types of events share the same basic attributes. However, in addition to the attributes they inherit from their raw events, active events can be configured with attributes that are specific to them (for details on active event attributes, see “Active Event Attributes” on page 49). By using the `eventbase_data1-data10` attributes, you can define additional attributes that will display content that is specific to your IT infrastructure.

Tip: To view the raw event table, right-click an event in one of the event tabs in the Information pane of the Map View and select **Event Log**.

This section contains the following topics:

- “Changing the Display Name of Additional Attributes” on page 39
- “Defining an Event Rule Action for Additional Attributes” on page 41

Changing the Display Name of Additional Attributes

There are two ways you can change the display name of the `eventbase_data1-data10` attributes.

You can:

- change the display name in the `eventbase.xml` file
- change the display name in the Class Browser

Tip: If you change the display names in the **eventbase.xml** file, then you need to activate the **OnlineDBCcreator.cmd** file for the changes to take effect. Bear in mind that this will delete all the data you have in the database and build a new one from scratch. If you change the display name in the Class Browser, you do not have to activate the **OnlineDBCcreator.cmd** file for the changes to take effect. It is therefore recommended to make the changes in the Class Browser.

To change the display name of the eventbase_data1-data10 attributes in the eventbase.xml file:

- 1 In the **eventbase.xml** file located in: **Mercury\<Mercury Application Mapping directory>\root\lib\server\db\classes\root\Event classes**, at the end of the **<ATTRIBUTE>** tag, add **DISPLAY_NAME="<name of data>"** to each data attribute whose display name you want to change.

For example:

```
<ATTRIBUTE ATTR_NAME="order" ATTR_TYPE="java.lang.Integer"
MANDATORY="true" SIZE="3" DISPLAY_NAME="Order"/>
```

- 2 For the changes to take effect, run the **OnlineDBCcreator.cmd** file (for more details, see “Building the Topology Database” on page 143. Remember that running the **OnlineDBCcreator.cmd** file will erase all the data in your database.

To change the display name of the eventbase_data1-data10 attributes in the Class Browser:

- 1 In the Class Browser, right-click **eventbase** in the Explorer pane and select **Edit Class** to open the Edit Class dialog box.
- 2 In the **Class Attributes** tab, select the required **eventbase_data** attribute.
- 3 Click **Edit** to open the Edit Attribute dialog box.
- 4 Make the required changes in the **Display Name** box and click **OK** to save your changes.

Defining an Event Rule Action for Additional Attributes

You can define the event rule's action for **eventbase_data1-data10** attributes.

To define an event rule action for additional attributes:

- 1** Select **Administration > Event Configuration**.
- 2** Click the **Event Rules** tab.
- 3** In the **Condition** section, click **Add** to open the Event Rule dialog box.
- 4** After you define the class and condition, click **Add** in the **Actions** section to open the Add New Action dialog box.
- 5** From the Available Actions list, select **SetActiveEventAttribute** and click **Next** to open the next dialog box.
- 6** In the **Value** box, enter the first data attribute that you want to be displayed.

If you do not want to start with **data1**, enter the first data attribute you want to use (for example: **activeevent_data4**). The values you enter in the discovery pattern start from the data attribute you specify.

- 7** Click **Next** to open the next Add New Action dialog box.
- 8** Enter the reference to the values you defined in the discovery pattern. For example, enter **eventmessage_param(1...5)**.
- 9** Click **Finish**.

Saving Active Events Using Event Utilities

Active events are not saved directly after their creation but are kept in the system's memory, due to performance considerations. To prevent the loss of all active events in case of a server crash, you can use the built-in utilities which automatically save active events to the database.

By default, active events are automatically saved when the following actions occur:

- ▶ the active events accumulate to a certain amount or a certain time interval has passed since the last automatic saving – for more details, see “Configuring the Automatic Saving of Active Events” on page 43.
- ▶ the server is started. When the server restarts after a crash, it automatically recalculates the raw events in the database and generates active events from the raw events.
- ▶ the server is closed using the **server_shutdown.cmd** file – for more details, see “Starting and Shutting Down the Mercury Application Mapping Server” on page 136.
- ▶ the **initActiveEvent.cmd** file is executed to delete active events from the database – for more details, see “Deleting Active Events from the Database” on page 44.
- ▶ the **hot_export.cmd** file is executed to export the database to an external file – for details, see “Exporting the Topology Database” on page 144.
- ▶ an object is removed from the database – for details, see “Event Rule Actions” on page 30.
- ▶ the **saveActiveEventsToDb.cmd** file is executed to manually save active events – for more details, see “Manually Saving Active Events” on page 44.

This section contains the following topics:

- ▶ “Configuring the Automatic Saving of Active Events” on page 43
- ▶ “Manually Saving Active Events” on page 44
- ▶ “Checking Object and Active Event Synchronization” on page 44
- ▶ “Deleting Active Events from the Database” on page 44

Configuring the Automatic Saving of Active Events

Active events are automatically saved when one or both pre-defined parameters are fulfilled – the number of accumulated events or a time interval that passes between each automatic saving.

The amount parameter—**ActiveEventBulkSize**—determines that events are automatically saved when they accumulate to a certain amount. This amount includes any creation, deletion or updating of an active event. You can change the default (10,000), if you want the system to automatically save larger or smaller numbers of active events.

The unit of the time parameter—**ActiveEventBulkTime**—is based on minutes. The system's default time is 60 minutes, meaning that every hour all active events in the system are saved automatically.

The amount parameter and the time parameter are calculated simultaneously. Therefore, the first parameter value that is fulfilled (either amount or time) causes an automatic saving after which both parameters are recalculated. The default parameter values are configurable, and you can change them in the **appilogConfig.properties** file.

To configure the automatic saving of active events:

- 1** Open the `\Mercury\<Mercury Application Mapping directory>\root\lib\server\appilogConfig.properties` file in a text editor.
- 2** Locate the **General stuff properties** section. These parameters determine automatic saving.
 - To change the number of active events that are automatically saved, change the value of the **ActiveEventBulkSize** parameter.
 - To change the time interval between each automatic saving, change the value (in minutes) of the **ActiveEventBulkTime** parameter.
- 3** Save the file.
- 4** For the changes to take effect, restart the server.

Manually Saving Active Events

Mercury Application Mapping enables you to view only active events related to objects that are part of displayed views. However, you can examine the current state of all active events in the system, you can manually save the active events to the database, and see all of them in the **ACTIVEEVENT** table.

To manually save active events:

Double-click the **saveActiveEventsToDb.cmd** file, located in:
`\Mercury\\scripts\utils.`

All current active events are saved to the **ACTIVEEVENT** table in the database.

Checking Object and Active Event Synchronization

You can check whether objects and their active events are synchronized. For example, if an object is in a Minor state and its related active event is in a Critical state, they are not synchronized.

To check object and active event synchronization:

Double-click the **checkSyncActiveEvent.cmd** file located in:
`\Mercury\\scripts\utils.`

The results of the synchronization check are displayed in the **events.log** file, located in: `\Mercury\\root\logs.`

Deleting Active Events from the Database

You can delete all active events from the database. The state of all objects returns to Normal.

To delete active events from the database:

Double-click the **initActiveEvent.cmd** file located in:
`\Mercury\\scripts\utils.`

Raw and Active Event Attributes

Active events are always created from raw events and therefore both types of events share the same basic attributes. However, in addition to the attributes they inherit from the raw events, active events possess other attributes that are specific to them.

This section includes the following topics:

- “Raw Event Attributes” on page 46
- “Internal Raw Events Attributes” on page 48
- “Active Event Attributes” on page 49

Raw Event Attributes

The following raw event attributes are displayed in Mercury Application Mapping. They include the raw event's display name and database name:

Attribute Display Name	Attribute Database Name	Definition
severity	eventbase_severity	The severity of the raw event (Normal, Warning, Minor, Major, or Critical). Each severity level is displayed in a different color.
eventide	eventbase_eventid	The type of raw event.
classname	eventbase_classname	The class name of the raw event's object.
createtime	root_createtime	The date and time that the raw event is written to the database.
message	eventbase_message	A textual description of the raw event.
usertime	eventbase_usertime	The date and time at which a change in the state of a managed object is discovered. This discovery causes the creation of a raw event. The date and time are recorded by the Probe Manager once it receives task results that create a raw event. If this attribute is null (for example, in the case of a raw event that is based on a user action) and the raw event generates an active event, the active event's attribute receives the value of the createtime attribute.
attributename	eventbase_attributename	For raw events of Collector Threshold type only. Indicates the name of the attribute that causes the event.
attributevalue	eventbase_attributevalue	For events of Collector Threshold type only. Indicates the value of the attribute that causes the event.

Attribute Display Name	Attribute Database Name	Definition
system	rawevent_system	(For future use) Differentiates between sources of data that are stored in one database. For example, data that is related to two customers can be managed in one database, by using different rawevent_system values for each customer. Currently, you should use "MAM" as the sole attribute value.
subsystem	rawevent_subsystem	Differentiates between Mercury sub-systems that deliver raw events to the application server. Currently, these attribute values and the categories of the eventbase_category attribute are the same. The discovery patterns use the "Collectors" category only.
parameter	rawevent_parameter	Dynamic parameters that can be sent as part of the raw event value. These parameters can be addressed and used by the Event Rules, which are defined in the Event Configuration dialog box.

Internal Raw Events Attributes

The following internal raw event attributes are not displayed in Mercury Application Mapping but are stored in the database:

Attribute Name	Definition
eventbase_category	<p>Differentiates between types of raw events that belong to one vendor. Use this attribute to sort raw events into categories to facilitate their management. Currently, there are 10 categories for the Mercury Application Mapping vendor:</p> <ul style="list-style-type: none"> 1- COLLECTORS 2-CORRELATION 3-ACTIVE_EVENT 4-EVENT_MANAGER 5-PQL 6-WORLD 7-MAP 8-PQL_MANAGER 9-COLLECTORS_TASKS 10-COLLECTORS_RESULTS <p>These categories are defined in the a_enumValues.xml file, located in: \Mercury\<Mercury Application Mapping root directory>\root\lib\server\db\create_enums.</p> <p>Discovery patterns use the Collectors category only.</p>
id	<p>The unique identifier of the raw event. This attribute value is automatically set by Mercury Application Mapping and should not be defined manually in the discovery patterns.</p>
root_container	<p>The unique identifier of the object related to the raw event. This attribute value is automatically set by Mercury Application Mapping and should not be defined manually in the discovery patterns.</p>

Active Event Attributes

The following table describes the specific attributes of Mercury Application Mapping active events, which appear in the Information pane.

Attribute Name	Description
label	The inclusive label of the object related to the event. Includes other objects that identify the selected object. For example, a disk's label includes the disk name and its host name.
isack	Indicates whether the event has been acknowledged. In addition to changing the acknowledged state from the shortcut menu, you can change it from this box. To acknowledge the event, select the check box. To reverse the acknowledgment, clear the check box.
ackby	By whom the event is acknowledged, or un-acknowledged after prior acknowledgment. This box is automatically populated once the event is acknowledged or un-acknowledged.
acktime	The time of the last acknowledgment or un-acknowledgment.
note	User notes. This box can be populated by right-clicking the event to open the Event Attribute dialog box. The user can enter notes in the Value cell of the Note attribute.
lastsystemtime	The last date and time when a related raw event is written to the database.
lastusertime	The last date and time at which a change that causes the creation of the related raw event is discovered.
counter	The number of times that the same active event is written to the database without any severity or message change.
createcounter	The number of times an event switches from an inactive state to an active one. The switch from active to inactive is performed by the Clear action, which deactivates the active event. As a result, the active event is removed from the Event Browsers. However, the event can still be restored by each of the actions that updates one of the active event's attributes.

Attribute Name	Description
replacecounter	The number of times that the same active event is written to the database with a severity change.
activetime	The last date and time that an event becomes active.
issuppress	Usually for correlation events. Determines whether the object's icon blinks once the event occurs, similarly to the effect of the acknowledgment action. (Both actions determine whether the icon blinks. One is performed automatically by Mercury Application Mapping, whereas the other is performed by the user. If only one of the actions applies to the object, the icon blinks.) The suppression action automatically defines, for all correlation rules, that objects affected by a root-cause object do not blink even though active events are associated with the object. For example, if a correlation rule is defined for the connection between a network object and its hosts, the suppression action prevents the network hosts blinking once the network is down.
supprestime	The date and time of the last suppression or un-suppression action.
data1-data10	Undefined attributes that enable you to add your own event attributes. Note: You define the data1-data10 values in the discovery pattern.
comment, removefromstatc aseidceventid	Not in use.
Event Status	The severity level of the object, as adjusted for the weight accorded to its significance in the system. (The significance weight is set in the Status Factor tab in the View Node Definition dialog box). For example, if the severity of the event is 5 and the status weight is defined at 100% , the object's Event Status is 5 ($5 \times 100 / 100$). If the weight is defined as 60% , the object's status is 3 ($5 \times 60 / 100$).

3

Discovery Management

This chapter gives a description of how to run the discovery process and how to activate and edit discovery patterns.

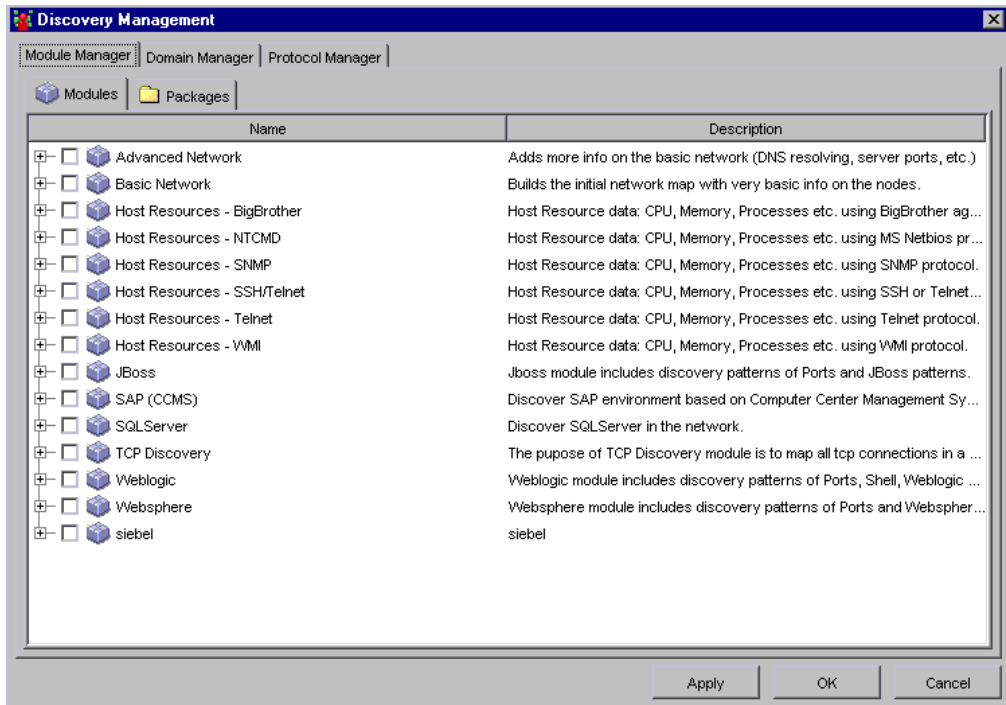
This chapter describes:	On page:
Understanding the Discovery Management Dialog Box	52
Configuring a Domain and a Discovery Scope	56
Configuring the Connection Data for a Protocol	59
Configuring a Module	63
Understanding the Pattern Editor	67
Editing a Pattern	69
Activating a Discovery Pattern	76
Protocol Definitions	77

About the Discovery Process

The discovery process is the mechanism that enables you to collect data about your system. It can discover such resources as applications, databases, network devices, different types of servers, and so on. Each discovered IT resource is then delivered and stored in the Mercury Universal CMDB where it is represented as a managed object. The Mercury Application Mapping discovery process is run by activating discovery patterns. For more details about discovery patterns and how to run the discovery process, see *Mercury Application Mapping Discovery Process Tutorial*.

Understanding the Discovery Management Dialog Box

You manage the discovery process by using the Discovery Management dialog box opened by selecting **Administration > Discovery Manager**.



The Discovery Management dialog box contains the following tabs:

- ▶ **Module Manager** – Enables you to activate discovery patterns and edit the XML files either in the XML file itself or in the Edit Pattern dialog box. The Module Manager includes the following tabs:
 - ▶ **Modules** – Displays a list of modules. Each module includes a list of all the patterns that are required for discovering a specific group of objects (for more details, see “Modules Tab” on page 53).
 - ▶ **Packages** – Displays a list of the deployed packages that contain discovery patterns (for more details, see “Packages Tab” on page 55).



Note: You can choose to activate patterns either from the Modules tab or from the Packages tab.

- ▶ **Domain Manager** – Enables you to define a new domain and the range of the net addresses to be discovered. For additional information on the Domain Manager, see “Configuring a Domain and a Discovery Scope” on page 56.
- ▶ **Protocol Manager** – Enables you to define the connection data for each protocol. For additional information on the Protocol Manager, see “Configuring the Connection Data for a Protocol” on page 59.

Modules Tab

The Modules tab contains a list of modules. Each module includes the patterns necessary to discover a specific group of objects.

The following table describes the icons in the Modules tab.

Icon	What it represents
	A module
	A pattern

The Modules tab has the following fields:

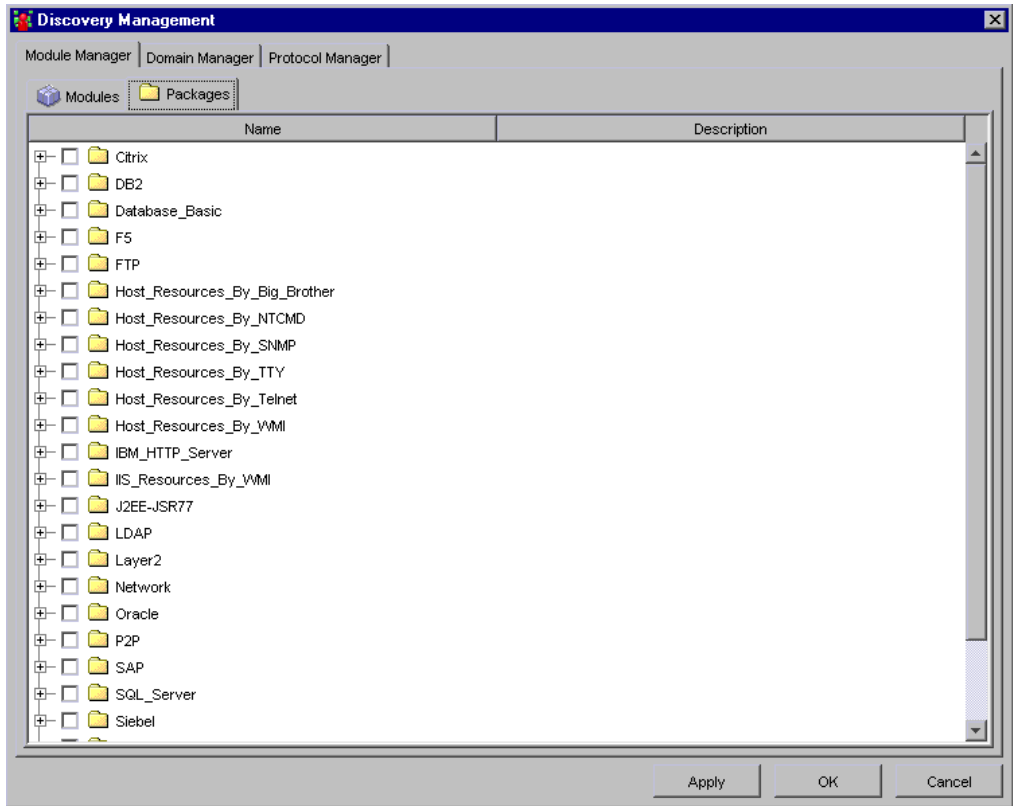
- ▶ **Name** – The name of the module and the name of the patterns included in the module when the module is in an expanded state.
- ▶ **Description** – The description of the module/pattern.

In the Modules tab you can:



- ▶ activate a Discovery Pattern in a module – for details, see “Activating a Discovery Pattern” on page 76
- ▶ edit a Pattern in a module – for details, see “Editing a Pattern” on page 69
- ▶ save a pattern to file – for details, see “Saving a Pattern to File” on page 76
- ▶ edit a module definition – for details, see “Editing a Module Definition” on page 65
- ▶ save a module definition to file – for details, see “Saving a Module Definition to File” on page 65
- ▶ export all the patterns in the module to file – for details, see “Exporting all the Patterns in the Module to File” on page 66

Packages Tab

The Packages tab lists all the deployed packages that contain discovery patterns. It enables you to activate one or more patterns from each package and also edit the discovery pattern.



The following table describes the icons in the **Packages** tab.

Icon	What it represents
	A deployed package
	A pattern

In the Packages tab you can:

- ▶ activate a discovery pattern in a package – for details, see “Activating a Discovery Pattern” on page 76
- ▶ edit a pattern in a package – for details, see “Editing a Pattern” on page 69
- ▶ save a pattern to file – for details, see “Saving a Pattern to File” on page 76

Configuring a Domain and a Discovery Scope

You define a new domain and a discovery scope using the Domain Manager tab.

This section contains the following topics:

- ▶ “Adding a New Domain and Configuring the Discovery Scope” on page 56
- ▶ “Rules for Defining an IP Address Range” on page 58

Adding a New Domain and Configuring the Discovery Scope

You can add a new domain and configure the discovery scope.

To add a new domain and configure the discovery scope:

- 1** Select **Administration > Discovery Management** to open the Discovery Management dialog box.
- 2** Click the **Domain Manager** tab.
- 3** Click **Add** in the **Domain** section to open the Add New Domain dialog box.
- 4** In the **Name** box, type the new domain name. The domain name definition refers to the name of the domain to be discovered.
- 5** (Optional) In the **Description** box, type the domain description.
- 6** Click **OK** to add the new domain to the **Domain** list and open the Add Range dialog box.
- 7** Enter an IP address range (for details, see “Rules for Defining an IP Address Range” on page 58).

- 8 Click **OK**. The range of net addresses you defined appears in the **Ranges** section.

To enter another IP address range, click the **Add** button in **Ranges** section and then repeat steps 7 to 8.

- 9 Click **OK** to save the changes you have made.

To delete a domain:

- 1 Select **Administration > Discovery Management** to open the Discovery Management dialog box.
- 2 Click the **Domain Manager** tab.
- 3 Select the required domain from the **Domain** list and click the **Delete** button in the **Domain** section.
- 4 Click **OK** to save the changes you have made.

To edit a domain's details:

- 1 Select **Administration > Discovery Management** to open the Discovery Management dialog box.
- 2 Click the **Domain Manager** tab.
- 3 Select the required domain from the **Domain** list.
- 4 Click the **Edit** button in the **Domain** section to open the Edit Domain dialog box.
- 5 Make the required changes.
- 6 Click **OK** to save the changes you have made.

To delete a range:

- 1 Select **Administration > Discovery Management** to open the Discovery Management dialog box.
- 2 Click the **Domain Manager** tab.
- 3 Select the domain range and click **Delete**.
- 4 Click **OK** to save the changes you have made.

To edit a domain's range:

- 1** Select **Administration > Discovery Management** to open the Discovery Management dialog box.
- 2** Click the **Domain Manager** tab.
- 3** Select the required domain from the **Ranges** section.
- 4** Click the **Edit** button in the **Ranges** section to open the Edit Range dialog box.
- 5** Make the required changes.
- 6** Click **OK** to save the changes you have made.

Rules for Defining an IP Address Range

The rules for defining an IP address range are as follows:

- ▶ the IP address range must have the following format:
start_ip_address – end_ip_address
For example: **10.0.64.0 - 10.0.64.57**
- ▶ the range can include a wildcard character (*) so that Mercury Application Mapping can match the range to more than one IP address. Mercury Application Mapping scans the system to find the IP addresses matching the range pattern you defined.
- ▶ an asterisk (*) represents any number in the range of 0-255.
- ▶ you can use a wildcard character (*) only in the lower bound IP address of the IP range pattern.

For example:

Valid	Not Valid
10.0.64.* - 10.0.64.10	10.0.64.10 - 10.0.64.*

- ▶ if you do use an asterisk (*), you do not need to enter a second IP address. For example, you can enter the range pattern **10.0.48.*** to cover the whole range from **10.0.48.0** to **10.0.48.255**.

- ▶ you can use more than one asterisk (*) in an IP address as long as they are used consecutively.

For example:

Valid	Not Valid
10.0.64.*	10.*.64.*
10.0.*.*	
10.*.*.*	

Note: The first number in an IP address cannot be substituted for an asterisk (*). For example, *.0.60.10 is not a valid range pattern.

- ▶ if you use an asterisk (*) in the lower bound IP address and also enter an upper bound IP address, the upper bound IP address is ignored. For example, if you enter the pattern 10.0.*.* - 10.0.20.30, the upper bound IP address is ignored. Since the asterisks (*) in the lower bound IP address cover a range wider than 20 and 30, 20 and 30 in the upper bound IP address are rendered irrelevant.

Configuring the Connection Data for a Protocol

You add connection data for each protocol included in the discovery process to the domain. The connection data can refer to a specific net address and/or to the entire net addresses range. When referring to the entire range, the net address value is DEFAULT. The default definitions relate to all IPs included in the defined range.

For a description of the connection data that needs to be defined for each protocol, see “Protocol Definitions” on page 77.

This section contains the following topics:

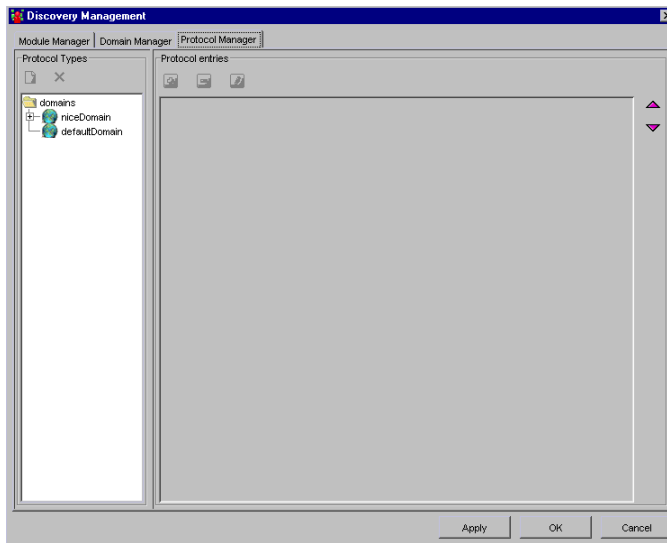
- “Defining the Connection Data for the Protocol you Added” on page 60
- “Deleting an Existing Protocol” on page 62
- “Deleting the Connection Details for an Existing Protocol” on page 62
- “Editing the Connection Details for an Existing Protocol” on page 63

Defining the Connection Data for the Protocol you Added

You can define the connection data for the protocol you added.

To define the connection data for the protocol you added:

- 1 Click **Administration > Discovery Manager > Protocol Manager** tab to open the following.

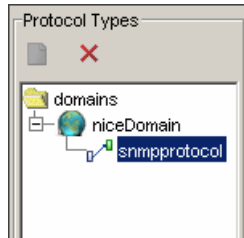


- 2 In the **Protocol Types** section, select the domain to which you want to add an instance of a protocol.



- 3 Click **Add new protocol type to selected domain**

- From the **Add Protocol** list, select the protocol you want to add. The protocol appears under the selected domain in the **Protocol Types** box, as illustrated below.



The Add Protocol Parameters page opens.

Note: If the domain appears red, it indicates that not all the required protocols for the discovery pattern were added as defined in the **Contract** tab in “Contract Tab” on page 69. To add the missing protocols, right-click the domain and add the required protocols.

- Select the added protocol and click the **Add new connection details for selected protocol type** button in the **Protocol Entries** section to add definitions to the protocol you selected.



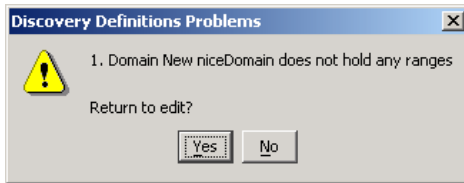
The Add Protocol Parameter dialog box opens and displays the list of attributes you need to define for the protocol you added.

- Define the protocol parameters as required and then click **OK**. The parameter values you have defined appear in the **Protocol entries** section.

Note: If the user has remove, create and update permissions for that protocol, the value in the Community column appears decrypted.

- Click **OK** and then **Save** in the Discovery Management dialog box to save the changes you have made.

If your discovery management definitions are incorrect or incomplete, you get a message specifying what the problem is, as seen in the following example:




8 Click:

- ▶ **Yes** to open the tab in which the issue has to be resolved
- ▶ **No** to save the changes and close the Discovery Management dialog box

Deleting an Existing Protocol

You can delete an existing protocol.


To delete an existing protocol:

-  **1** In the Protocol Manager tab in the Discovery Management dialog box select the required protocol and click the **Remove protocol type from selected domain** button.
- 2** Click **OK** and then **Save** in the Discovery Management dialog box to save the changes you have made.

Deleting the Connection Details for an Existing Protocol

You can delete the connection details for an existing protocol.


To delete the connection details for an existing protocol:

-  **1** In the Protocol Manager tab in the Discovery Management dialog box select the entry that you want to delete in the **Protocol entries** section click the **Remove selected connection details for selected protocol type** button.
- 2** Click **OK** and then **Save** in the Discovery Management dialog box to save the changes you have made.

Editing the Connection Details for an Existing Protocol

You can edit the connection details for an existing protocol.

To edit the connection details for an existing protocol:

- 1** In the Protocol Manager tab in the Discovery Management dialog box select the entry that you want to edit in the **Protocol entries** section.
-  **2** Click the **Edit selected connection details for selected protocol type** button.
- 3** Edit the details as required in the Edit Protocol Parameter dialog box that opens.
- 4** Click **OK** and then **Save** in the Discovery Management dialog box to save the changes you have made.

Configuring a Module

This section contains the following topics:

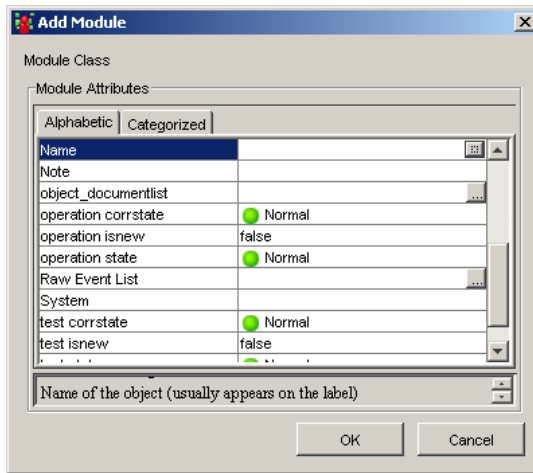
- “Creating a New Module” on page 64
- “Editing a Module Definition” on page 65
- “Saving a Module Definition to File” on page 65
- “Exporting all the Patterns in the Module to File” on page 66

Creating a New Module

This section describes how to create a new module.

To create a new module:

- 1 Right-click anywhere in the **Modules** tab and select **Create new module**. The Module Class dialog box opens.



- 2 Define the following attributes:



- Click the button at the right end of the **Name** row and enter the name of the module you want to create.



- (Optional) Click inside the **discoverymodule_description** row at the right and click the button.

- Enter a description of the module in the **discoverymodule_description** dialog box that opens.



- To add a discovery pattern, click the button at the right end of the **discoverymodule_patterns** row. The Edit list of values dialog box opens.



- Click the **Add** button.



- Click inside the row and click the button to open the **discoverymodule_patterns** dialog box.

- Type the name of the pattern you want included in the module.

- Click **OK**.

Note: You can add as many discovery patterns as required.

- 3 Click **OK** in the Add Module dialog box to add the module you have created in the **Modules** tab.

Editing a Module Definition

This section describes how to edit a module definition.

To edit the module definition:

- 1 Right-click the module and select **Edit module definition** to open the Module Attributes dialog box.

The Module Attributes dialog box lists the classes contained in this module and their attribute values.

- ▶ Click the **Categorized** tab to group the classes according to the categories defined in the Class Browser.
- ▶ Click the **Alphabetic** tab to list the classes in alphabetical order.

- 2 If you are using the Categorized tab, click the **Expand** button to view all the classes contained in this module.
- 3 In both tabs, click in the column to the right of the class you want to edit. A button appears if the row is editable.
- 4 Click the button and edit the class attribute as desired.
- 5 Click **OK** to save the changes you have made.

Saving a Module Definition to File

This section describes how to save the module definition in XML format.

To save the module definition to file:

- 1 Right-click the desired module and select **Export module definition to file** to open the **Save** dialog box.
- 2 Browse to the location where you want to save the module definition.

- 3 Click **Save**.

Note: You can only save the module definition in XML format.

Exporting all the Patterns in the Module to File

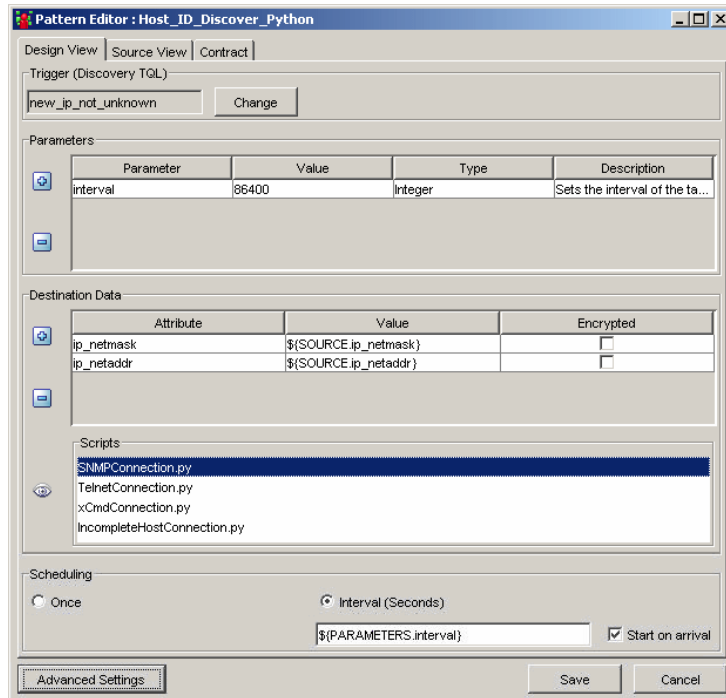
This section describes how to save all the patterns in the module to file.

To export all the patterns in the module to file:

- 1 Right-click the desired module and select **Export all patterns** to open the Save dialog box.
- 2 Browse to the location where you want to save all the patterns.
- 3 Click **Save**.

Understanding the Pattern Editor

The Pattern Editor dialog box is as follows:



The Pattern Editor dialog box contains the following tabs:

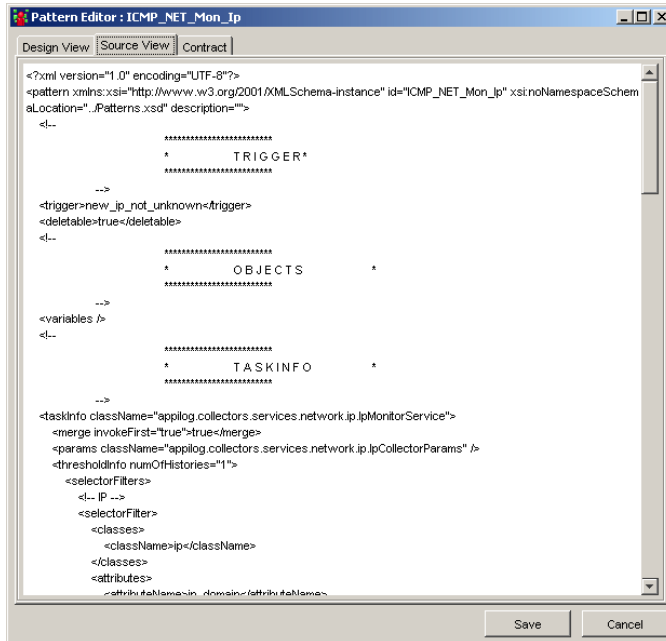
- “Design View Tab” on page 68
- “Source View Tab” on page 68
- “Contract Tab” on page 69

Design View Tab

You can design and edit the discovery pattern in the Design View tab, as seen in “Editing a Pattern” on page 69.

Source View Tab

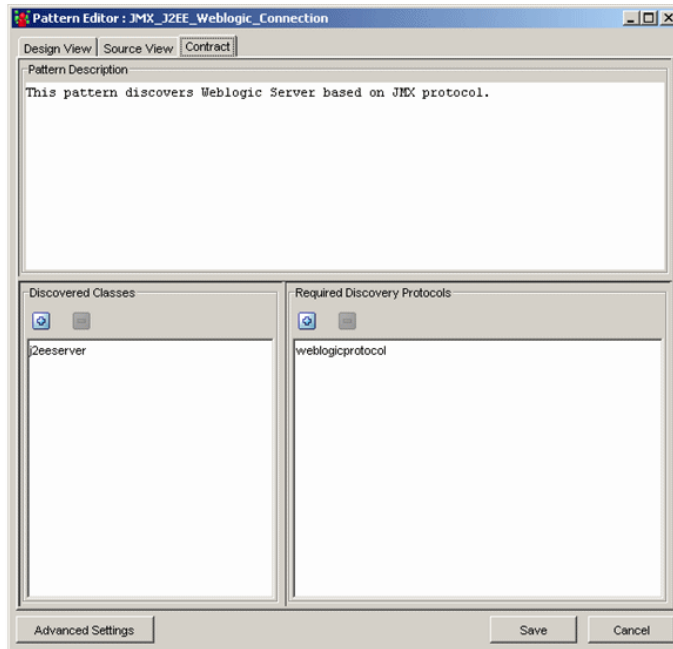
The Source View tab displays the discovery pattern in XML format, which can be edited.



Contract Tab

Enables you to describe:

- The classes that are loaded (discovered) during the discovery process.
- The protocols that are required to perform the discovery process.



Editing a Pattern

You can edit a pattern in the Modules tab or in the Packages tab by accessing the Pattern Editor. The Pattern Editor allows you to either edit the pattern in XML format in the Source View tab or in the Design View tab.

You can then:

- Design and edit a discovery pattern
- Display the discovery pattern in XML format
- Describe the classes that are loaded during the discovery process and the protocols that are required for the discovery process

This section includes the following topics:

- “Designing a Discovery Pattern” on page 70
- “Editing the Discovery Pattern in the Source View Tab” on page 74
- “Defining the Discovery Pattern” on page 75

Designing a Discovery Pattern

You can design a discovery pattern.

To design a discovery pattern:

- 1** Right-click the pattern you want to edit in the Packages or in the Modules tab and click **Edit Pattern** or double-click the pattern to open the Pattern Editor.
- 2** In the **Trigger (Discovery TQL)** section, click the **Change** button to select the TQL that serves as the trigger that invokes the discovery pattern’s task.
- 3** In the **Parameters** section, define the following parameter values
 - **Parameter name** – the name of the parameter
 - **Value** – the value you want to assign to the attribute
 - **Type** – (optional) the attribute type
 - **Description** – (optional) a description of the parameter

Note: Each row represents the definitions for one parameter.



- 4** To define another pattern parameter, click the add button. Another row of parameter attribute definitions appears. Configure the parameters according to the list above.



- 5** To delete a pattern parameter, select the parameter you want to delete and click the minus button.
- 6** In the **Destination Data** section, you define the information that is needed to perform a discovery task on a specific object. The information that is defined is passed on to the destination queried in the discovery task.

To configure the destination data, do the following:

- define the destination data attributes according to the following table:

Attribute	Description
Attribute name	The name of the attribute.
Value	<p>The attribute value. Variables are written using the following syntax:</p> <p><code>\${VARIABLE_NAME.attributeName}</code></p> <p>where <VARIABLE_NAME > can either be one of three predefined variables:</p> <ul style="list-style-type: none"> ➤ Source – Refers to the object that functions as the task’s trigger. ➤ Host – Host in which the source object is contained. ➤ Parameters – This variable refers to the parameter defined in the Parameter section as described above. <p>or a variable that you have created (see “Source View Tab” on page 68)</p> <p>For example:</p> <p><code>\${SOURCE.network_netaddr}</code></p> <p>indicates that the trigger object is a network.</p>
Encrypted	Select this check box if the field is defined as a Password type in the Mercury Universal CMDB.



- to define another attribute, click the **Add** sign. and modify the attribute according to the table above.



- to delete an existing attribute, select the attribute you want to delete and click the **Delete** sign.

7 In the **Scripts** section, you can view the Python scripts that are run by the selected pattern. To view the Python script, do the following:

- Select the Python script you want to view



- ▶ Click the **View Script** icon to open the Script Editor window:

```
Script Editor : JMX_I2EE_Websphere.py
import traceback
import string, sys
from java.lang import System
from applog.common.system.types.vectors import ObjectStateHolderVector
from applog.common.system.types import ObjectStateHolder
from applog.common.system.types import AttributeStateHolder
from applog.common.utils.parser import OperatorParser
from applog.collectors.services.dynamic.agents import WebsphereAgent
from applog.collectors.services.dynamic.agents import AgentConstants
from java.lang import Integer
from java.util import *
from javax.management import ObjectName
from applog.collectors.util import HostKeyUtil

# log4j stuff
from org.apache.log4j import Category

# get the logger instance for this script
logger = Category.getInstance("PATTERNS_DEBUG")
#logger = Framework.getLogger()

if logger.isInfoEnabled():
    logger.info("Start JMX_I2EE_Websphere.py")

#####
# a convenient print debug method
```

Note: You cannot edit the script in the Pattern Editor.

- ▶ Click **Close** to close the Script Editor window

8 In the **Scheduling** section, do the following:

- ▶ Select **Once** if you want to activate the discovery task only once.
- ▶ Select **Interval** to activate the discovery task at predefined intervals. The interval is measured in seconds.

For example, if you want to activate the discovery task every 60 seconds, You can either type the integer 60 or use the following format:

`${PARAMETERS.interval}`

if you have defined a parameter in the **Parameters** section using the following values: **Parameter name:** interval and **Value:** 60

Parameters			
Parameter	Value	Type	Description
interval	60	Integer	Sets the interval of the ta...

- ▶ Select **Start on Arrival** if you want to activate the discovery task when it reaches the Probe Manager. This option is only enabled if you have selected the **Interval** option.
- 9** To define advanced settings for the pattern, click **Advanced Settings** to open the Advanced Settings dialog box.
- 10** In the **Task Management** section, do the following:
- ▶ Select **Merge** if you want a task to include several destinations rather than only one destination per task. The default is selected.
 - ▶ Select **EnforceDispatch** if you want to invoke a task for objects whose domain is not included in the discovery scope range you define in the Domain Manager tab (see on “Configuring a Domain and a Discovery Scope” on page 56). The default is not selected.
 - ▶ If an object that acted as a trigger for a discovery pattern was deleted from the database, select **Deletable** if you want the discovery task it activated to be deleted as well. The default is selected.
- 11** When there is more than one Probe Manager in a domain, the Probe Gateway randomly selects the IP address of one of them to run the discovery tasks. If you want to specify which Probe Manager you want to run the task, do the following:
- ▶ In the **Probe Manager Selection** section, select **Override Default Selection Process**.
 - ▶ In the **Probe Manager Address** box, enter the IP address of the Probe Manager you want to run the discovery task.
- 12** In the **Discovery Service** section, the **Class name** box contains the name of the Java class that is activated once the task is invoked. If you want to edit the Java class name, do the following:
- ▶ Select **Edit**.
 - ▶ Edit the **Class name** box as required.
- 13** In the **Result Grouping** section:
- ▶ Select **Group Results** if you want to store discovery results in the Probe Manager before being sent to the Mercury Application Mapping server. This option is designed for discovery patterns that are of the type Listeners only.

- In the **Grouping Interval (Seconds)** box, type the value that indicates how long discovery results are stored in the Probe Manager before being transferred to the Mercury Application Mapping server.
- In the **Group Max Objects** box, specify the number of objects that should accumulate in the Probe Manager before being transferred to the Mercury Application Mapping server.

Note: If you entered a value in both fields, Mercury Application Mapping applies whichever occurs first.

- If you do not select **Group Results**, all discovery results are immediately sent to the Mercury Application Mapping server.

Editing the Discovery Pattern in the Source View Tab

The Source View tab displays the discovery pattern in XML format, which can be edited.

To edit the discovery pattern in the Source View tab:

- 1** Right-click the pattern you want to edit in the Packages or in the Modules tab and click **Edit Pattern** or double-click the pattern to open the Pattern Editor.
- 2** Click the **Source View** tab.
- 3** Make the required changes.
- 4** Click **Save** to save the changes you have made.

Note: You can copy and paste the text to work in any text editor and then paste it back in the Pattern Editor.

Defining the Discovery Pattern

You define a discovery pattern by specifying the classes the pattern will discover and the protocols needed to perform the discovery.

To define a discovery pattern:

- 1** Right-click the pattern you want to edit in the Packages or in the Modules tab and click **Edit Pattern** or double-click the pattern to open the Pattern Editor.
- 2** Click the **Contract** tab.
- 3** In the **Pattern Description** box, type a description of the discovery pattern.
- 4** To define which classes the pattern discovers, do the following:



- ▶ In the **Discovered Classes** box, click the **Plus** button to display the classes in the Class Model.
- ▶ Select the class or classes you want the pattern to discover.

To delete an existing class from the **Discovered Classes** box, select the class you want to delete and click the **Minus** button.



- ▶ Click **OK** to save the changes you have made.

- 5** To define which protocols the pattern requires for the discovery task, do the following:



- ▶ In the **Required Discovery Protocols** box, click the **Plus** button to open the Add Required Protocol dialog box.
- ▶ From the **Choose Protocol Type** list, select the required protocol.

You can also delete an existing protocol from the **Required Discovery Protocols** box, by selecting the protocol you want to delete and clicking the **Minus** button.



- ▶ Click **OK** to save the changes you have made.
- ▶ Click **Save** to save your definitions.

Activating a Discovery Pattern

You can activate a discovery pattern in a package or in a module. You can then save the pattern to a file.

This section contains the following topics:

- ▶ “Activating a Discovery Pattern” on page 76
- ▶ “Saving a Pattern to File” on page 76

Activating a Discovery Pattern

You can activate a discovery pattern in a package or in a module. You can activate either all the discovery patterns or some of them.

To activate all the discovery patterns in a module:

- 1** Select **Administration > Discovery Manager** and click the **Modules** tab in the Discovery Management dialog box.
- 2** You can either select the check box to the left of the module to activate all the patterns in the module or you can select individual patterns.
- 3** Click **Save** to activate the selected pattern(s).

To activate a discovery pattern in a package:

- 1** Select **Administration > Discovery Manager** and click the **Packages** tab in the Discovery Management dialog box.
- 2** You can either select the check box to the left of the deployed package, which automatically activates all the patterns in the package or you can select individual patterns.
- 3** Click **Save** to activate the selected pattern(s).

Saving a Pattern to File

You can save a pattern to file from either the **Packages** or **Modules** tab.

To save a pattern to file:

- 1** Right-click the desired pattern and select **Export pattern to file** to open the **Save** dialog box.

- 2 Browse to the location where you want to save the pattern.
- 3 Click **Save**.

Protocol Definitions

This section contains the definitions that are needed for Mercury Application Mapping protocols.

This section includes the following topics:

- “snmpprotocol” on page 78
- “sqlpprotocol” on page 78
- “wmiprotocol” on page 79
- “weblogicprotocol” on page 80
- “jbossprotocol” on page 80
- “telnetprotocol” on page 81
- “ftpprotocol” on page 82
- “sshprotocol” on page 82
- “siebelgtwyprotocol” on page 83
- “sapprotocol” on page 84
- “oracleprotocol” on page 85
- “ibmhttpserverprotocol” on page 85
- “websphere” on page 86
- “ntadminprotocol (pstools/xcmd)” on page 87

snmpprotocol

The protocol's parameters are as follows:

Parameter	Description
Community	The password used for authentication
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the SNMP agent
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number on which the SNMP agent listens
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
Retry	The number of times the Probe Manager tries to connect to the SNMP agent. If the number is exceeded, the Probe Manager stops attempting to make the connection.

sqlpprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the database
Database Name	The database name
Database SID (Oracle, DB2)	The database SID
Database Type	The database type, such as Oracle and Microsoft SQLServer

Parameter	Description
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number on which the database listens
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password

wmiprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the WMI agent
Network Address	The discovered IP net address or the net address range
Note	A textual message
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password
WMI Domain	The Microsoft domain name

weblogicprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the WebLogic application server
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password

jbossprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the JBOSS application server
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number

Parameter	Description
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password

telnetprotocol

The protocol's parameters are as follows:

Parameter	Description
Admin password (UNIX only)	In some UNIX systems, when connecting with a user name that is not "root," (that is, not administrator), and the connection data includes a password for the administrator user, the Probe Manager automatically switches users and tries to connect as the administrator user. (The automatic switch from a regular user to the administrator user is done through the su-root command). Therefore, if you want the Probe Manager to perform administrative actions, add the administrator user password to the Telnet definitions.
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the remote machine
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number
User Name	The user name

Parameter	Description
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Password	The user password

ftpprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the FTP server
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number
User Name	The user name
User Password	The user password

sshprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the remote machine.
Network Address	The discovered IP net address or the net address range
Note	A textual message

Parameter	Description
Port Number	The port number
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password

siebelgtwyprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the Siebel gateway
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
Siebel Site Name	The name of the Siebel site.
svrvmgrpath	The directory in which the svr.exe file is located

Parameter	Description
User Name	The user name
User Password	The user password

sapprotocol

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the SAP server.
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
Sap Client	An independent unit within the R/3 system, which is identified by three-digit number.
Sap Router String	A string that contains the host and port of the SAP router
Sap System Number	An unique identifier of SAP system
User Name	The user name
User Password	The user password

oracleprotocol

The protocol's parameters are as follows:

Parameter	Description
Network Address	The discovered IP net address or the net address range
Note	A textual message
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password

ibmhttpserverprotocol

The protocol's parameters are as follows:

Parameter	Description
Admin Console Port	The port on which the admin console is listening
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the IBM HTTP server.
Install Root Dir	Directory in which the IBM HTTP server is located.
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number

Parameter	Description
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
User Name	The user name
User Password	The user password

websphere

The protocol's parameters are as follows:

Parameter	Description
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the WebSphere server.
Key Store	The location of SSL key store file
Key Store Password	The SSL key store password
Network Address	The discovered IP net address or the net address range
Note	A textual message
Port Number	The port number
Protocol Index	Indicates the order in which protocol instances will be used to make a connection attempt. The lower the index is the higher the priority it has. The default is 9999. If you do not change the default, this protocol instance will be used last.
Trust Store	The location of SSL trust store file
Trust Store Password	The SSL trust store password
User Name	The user name
User Password	The user password

ntadminprotocol (pstools/xcmd)

The following protocols enable you to run commands on remote Windows workstations. In each command, a new connection should be established.

Parameter	Description
Admin password (UNIX only)	In some UNIX systems, when connecting with a user name that is not "root," (that is, not Administrator), and the connection data includes a password for the Administrator user, the Probe Manager automatically switches users and tries to connect as the administrator user. Therefore, if you want the Probe Manager to perform administrative actions, add the administrator user password to the Telnet definitions.
Connection Timeout	Timeout in milliseconds after which the Probe Manager stops trying to connect to the remote machine
Network Address	The discovered IP net address or the net address range
Port Number	The port number
User Name	The user name
User Password	The user password

4

Discovery Process Configuration

This chapter describes the configuration parameters and procedures of the discovery process.

This chapter describes:	On page:
Discovery Process Configuration Management	90
The Probe Gateway and Probe Manager Configuration File	91
Discovery Process Configuration Files	103
The Discovery System Directory Structure	110

About the Discovery Process Settings

This chapter describes the definition and parameter files needed for configuring the discovery process and a description of the content and use of the discovery process folders and files for each of the discovery components.

Discovery Process Configuration Management

The `\Mercury\<Mercury Application Mapping root directory>\root\lib\collectors\discoveryManager\serverData` folder includes files that contain definitions and parameters that mark out the discovery scope and provide necessary data, such as passwords, needed for the operation of the discovery process. This folder also includes other utilities, such as standard conversion lists, that convert the raw data (collected through the discovery process) into information.

The definitions and utilities are needed for the operation of the Probe Gateway and Probe Manager. The original files are saved in the Discovery Manager. When the Probe Gateway is installed, these files are written to the **serverData** folder. They are then distributed to the Probe Managers where they are saved to the local **serverData** folders.

The Probe Gateway initiates requests to the Discovery Manager for updated files on a regular basis. If changes have occurred in the files, the Probe Gateway receives updated copies of the **serverData** files and distributes them to the Probe Managers. Any changes to the discovery process definitions, such as a password change, are made in one place only—on the Mercury Application Mapping server—and are automatically updated in the other discovery components.

The files that are stored in the `serverData` folder are described in XML files (for details, see “Discovery Process Configuration Files” on page 103).

The Probe Gateway and Probe Manager Configuration File

Configuration parameters required by the discovery system for the discovery process are defined in the **appilog-remote.properties** file. The file includes parameters for the Probe Gateway, the Probe Manager, and the discovery methods.

The **appilog-remote.properties** file is located in:

```
\Mercury\<Mercury Application Mapping root directory>\root\lib  
\collectors.
```

Note: If you update **appilog-remote.properties** parameters, you must restart the Probe Manager and Probe Gateway.

This section contains the following topics:

- “Configuring the Configuration Parameters” on page 92
- “Server and Probe Gateway Connections” on page 92
- “Probe Gateway and Probe Manager Domains” on page 93
- “Connection Protocol Ports” on page 93
- “General Configurations” on page 95
- “Thread Pools” on page 95
- “Probe Gateway General Definitions” on page 97
- “Time Intervals for the Probe Gateway and Probe Manager Procedures” on page 98
- “Probe Manager General Definitions” on page 101

Configuring the Configuration Parameters

You can configure the parameters for the Probe Gateway, Probe Manager, and discovery methods

To configure the configuration parameters:

- 1 Open `apilog-remote.properties` in a text editor.
- 2 Edit the configuration parameters as described in the following sections.
- 3 Save the file.

Server and Probe Gateway Connections

The Server section contains parameters that are needed for the connection between the server and the Probe Gateway. This connection can be established through either HTTP or RMI protocol. During the installation process you determine which type of connection protocol is to be used (for details, see “Installing the Probe Gateway”, in *Mercury Application Mapping Installation Guide*). Following the installation, you can change the protocol definitions.

Parameters	Description
<code>serverIP</code>	IP address of the Mercury Application Mapping server
<code>serverPort = 7001</code>	Port number of the Mercury Application Mapping server
<code>serverPortHttps</code>	Port number of the Mercury Application Mapping server when using the HTTPS protocol
<code>serverDomain = niceDomain</code>	The domain in which the Mercury Application Mapping server sits
<code>server.webApp.name = mam</code>	The name of the Mercury Application Mapping web application

Parameters	Description
<code>server.download.servlet.name = downloader</code>	Internal servlet definitions for an HTTP communication between the Mercury Application Mapping server and the Probe Gateway Note: Do not change internal servlet definitions.
<code>server.collectors.servlet.name = collectors</code>	
<code>server.collectors.utilities.servlet.name = collectorsUtilities</code>	
<code>appilog.log.interval = 20</code>	Frequency with which the Probe Gateway and the Probe Manager send logs to the probe_hb.log and local_hb.log files (in seconds) regarding their state of health

Probe Gateway and Probe Manager Domains

The following parameters determine the Probe Gateway and Probe Manager Domains.

Parameters	Description
<code>appilog.collectors.domain = niceDomain</code>	Probe Gateway domain.
<code>appilog.collectors.local.ip</code>	Probe Manager domain.
<code>appilog.collectors.probe.ip</code>	The IP address that the Probe Gateway is running on.

Connection Protocol Ports

The following parameters determine the connection protocol ports.

Parameters	Description
<code>appilog.collectors.rmi.port = 1099</code>	RMI port between the Probe Gateway and Probe Manager. The connection between the Probe Gateway and Probe Manager is made through the RMI protocol, whereas the connection between the Probe Gateway and the server is made either through RMI or HTTP.
<code>appilog.collectors.local.html.port=8082</code>	HTTP port of the Probe Manager

Parameters	Description
<p>appilog.collectors.probe.html.port=8083</p>	<p>HTTP port of the Probe Gateway</p>
<p>appilog.collectors. ProbeUseSpecificRMIPortFrom = 1199</p>	<p>The port of communication between the Probe Gateway and the Probe Manager. If this port is already being used, then the first of the ten ports above it, 1200 to 1209, that is available, will be used. Both the Probe Gateway and the Probe Manager can be accessed through the HTTP protocol which channels HTML requests. You can view the HTML adapter through your browser. To view it, enter the following URL: <code>http://<server IP>:<HTTP port></code> where:</p> <p><server IP> is the IP address of the Probe Manager or the Probe Gateway</p> <p><HTTP port> is the HTTP port of the Probe Manager or the Probe Gateway</p>

General Configurations

The following parameters control the general configuration.

Parameters	Description
<code>appilog.collectors.probeLocalUnion = false</code>	If set to yes, then the Probe Gateway and the Probe Manager will as a single process.
<code>appilog.collectors.sendCustodialLinksToServer = false</code>	When you activate a discovery pattern, the objects that are discovered are sent to the Mercury Application Mapping server. If this attribute is set to true, then a link is created for each discovered object in the Mercury Application Mapping server. This enables you to know which pattern created or update which discovered object.

Thread Pools

The Probe Gateway and Probe Manager use predefined thread pools for efficient performance.

Parameters	Description
<code>pools = probeGeneral, localServices, localGeneral, probeDefault, localDefault, patrolPool</code>	Do not change the thread pool names.
<code>patrolPool</code>	Used for discovery methods that are related to BMC PATROL
<code>probeGeneral</code>	Performs other activities not included in the probeTaskResults pool, such as delivering task requests to the Probe Manager, managing the connection with the server, and so forth
<code>localServices</code>	Manages discovery methods
<code>localGeneral</code>	Performs other activities that are not included in the localServices pool, such as verifying that task results have been sent to the Probe Gateway

Parameters	Description
probeDefault	Used when an unidentified Probe Gateway pool thread is requested
localDefault	Used when an unidentified Probe Manager pool thread is requested

Each defined pool has a minimum and maximum number of threads it can use:

patrolPool.minPoolSize=0	localGeneral.minPoolSize=5
patrolPool.maxPoolSize=40	localGeneral.maxPoolSize=15
probeGeneral.minPoolSize=10	probeDefault.minPoolSize = 0
probeGeneral.maxPoolSize=20	probeDefault.maxPoolSize = 2
localServices.minPoolSize=15	localDefault.minPoolSize = 0
localServices.maxPoolSize=15	localDefault.maxPoolSize = 2

Note: If the Probe Gateway uses the maximum number of threads that are allocated to it (a situation which often occurs when using the HTML port), it is recommended to allocate additional threads if there are available CPU resources.

Probe Gateway General Definitions

The following parameters determine the Probe Gateway general definition.

Parameters	Description
appilog.agent.probe.protocol = HTTP	The protocol through which the Probe Gateway speaks with the Mercury Application Mapping server.
appilog.agent.probe.serverResponseThresholdTime = 60	The maximum time (measured in seconds) the Probe Gateway waits to get a response from the Mercury Application Mapping server. If the server does not respond within this time, the Probe Gateway stops sending the Mercury Application Mapping server the discovery results. In such a case, the Probe Gateway stores the discovery results in its local database.
appilog.agent.probe.initialConnection = 5	The initial number of open connections between the Probe Gateway and its main repository, that is, the database folder (for details on this folder, see “The probeGateway Folder” on page 112).
appilog.agent.probe.maxConnection = 20	The maximum number of open connections between the Probe Gateway and its main repository
appilog.agent.probe.saveTaskResults = false	If set to true, all the discovery results will be kept in the Probe Gateway’s local database
appilog.agent.Probe.NumOfResultsSenders = 5	For HTTP communication mode only The maximum number of parallel connections that the Probe Gateway can open for sending task results to the server.
appilog.agent.Probe.MaxBulkOfTasks = 100	For HTTP communication mode only The maximum number of task requests that can be delivered in one bulk from the server.

Parameters	Description
appilog.agent.Probe. MaxSecondsToWaitForTasks = 20	<p>When the Probe Gateway receives task requests from the Server, it accumulates the task requests that are already available and the task requests that are sent during the period determined by the maximum waiting time (in seconds) for the task requests specified by this parameter</p> <p>When this time is over, the Probe Gateway delivers the task requests to the Probe Manager that have accumulated on the Server.</p>
appilog.agent.Probe. MaxTaskResultsToSave = 2000	<p>The maximum number of task results that can be stored in the Probe Gateway main repository, that is, the database folder</p>

Time Intervals for the Probe Gateway and Probe Manager Procedures

The following parameters determine the time intervals (in milliseconds) of procedures performed by the Probe Gateway and the Probe Manager.

Parameters	Description
appilog.agent.probe.retrieveServerData. interval=60000	<p>The Probe Gateway initiates requests to the Server for updated configuration files (the files stored in the serverData folder).</p> <p>This parameter determines the time interval between each update request.</p>
appilog.agent.probe.reconnection. interval=30000	<p>The Probe Gateway checks the connection to its Probe Manager(s) on a regular basis.</p> <p>This parameter determines the time interval between each connection check.</p>

Parameters	Description
appilog.agent.probe.retrieveTasksFromServer.interval=30000	<p>In HTTP communication mode, the Probe Gateway takes the accumulated task requests from the Server on a regular basis.</p> <p>This parameter determines the time interval between each request poll. (This procedure occurs before the calculation of the MaxSecondsToWaitForTasks parameter, as described in “Probe Gateway General Definitions” on page 97.)</p>
appilog.agent.probe.retrieveTasksFromDB.interval=60000	<p>The Probe Gateway checks whether any non-delivered task requests in the main repository need to be sent to the Probe Manager.</p> <p>This parameter determines the time interval between each repository check.</p>
appilog.agent.probe.retrieveTaskResultsFromDB.interval=60000	<p>The Probe Gateway checks whether any non-delivered task results in the main repository need to be sent to the Server.</p> <p>This parameter determines the time interval between each repository check.</p>
appilog.agent.local.processGuarantee.interval = 20000	<p>The Probe Manager checks whether any non-delivered task results in the cache need to be sent to the Probe Gateway.</p> <p>This parameter determines the time interval between each cache check.</p>
appilog.agent.local.processGrouping.interval = 2000	Do not change the value of this parameter
appilog.agent.local.writeResultToLog.interval = 1000000	<p>In the Probe Manager, task results that are not delivered for more than a certain period of time are registered in the unSentTaskResults.log file and an event message is sent to the Server.</p> <p>This parameter determines the time interval that needs to pass before the non-delivered results are registered as not sent.</p>

Parameters	Description
<p>appilog.agent.local.sendAgainResultToProbe.interval = 120000</p>	<p>The Probe Manager sends task results to the Probe Gateway, but might not receive a confirmation message.</p> <p>In these cases, the Probe Manager resends the task results after a certain time interval. This parameter determines the time interval that needs to pass before the results are resent to the Probe Gateway.</p>
<p>appilog.agent.local.sendTopologyToProbe.interval = 45000</p>	<p>The Probe Manager regularly sends administrative information regarding the state of running tasks and discovery methods to the Probe Gateway.</p> <p>This parameter determines the time interval that passes between each information delivery.</p>
<p>appilog.agent.local.saveRepository = 60000</p>	<p>The Probe Manager regularly saves task results that are not received by the Probe Gateway due to connection problems.</p> <p>These task results are saved to the localResults folder in the unsentTaskResults.dat file, from which they can be retrieved when the connection problem is resolved. This parameter determines the time interval that needs to pass before the results are saved in the localResults folder.</p>
<p>appilog.agent.probe.checkIfCanSendResults.interval = 120000</p>	<p>The Probe Gateway regularly checks whether the Server is ready to receive task results.</p> <p>If a negative answer is returned, the Probe Gateway stops sending task results and saves them in its main repository, until a positive answer is sent from the Server. This parameter determines the time interval that passes between each availability check.</p>

Parameters	Description
<code>appilog.agent.probe.checkIfDeleteResults.interval = 900000</code>	The Probe Gateway regularly checks its repository to verify that the number of accumulated task results do not exceed a certain maximum number. This parameter determines the time interval that passes between each result check.
<code>appilog.agent.Probe.connectToJms.interval = 300000</code>	For future use

Probe Manager General Definitions

The following parameters control the number of threads that send the results to the probe.

Parameters	Description
<code>appilog.agent.local.isRemote = true</code>	Do not change the value of this attribute
<code>appilog.agent.Local.NumOfResultsSenders = 3</code>	The number of threads that are sending results from the Probe Manager to the Probe Gateway
<code>appilog.agent.local.initialThreads= 30</code>	Determines the initial number of threads that is allocated to all discovery methods Note: Discovery methods operate in a linear way, that is, they query their destination objects one after the other. However, thread pools can be allocated to the discovery methods to enable them to query their destination objects in parallel.
<code>appilog.agent.local.maxThreads= 100</code>	Maximum number of threads that is allocated to all discovery methods
<code>appilog.agent.local.threadsPerService = 8</code>	The number of threads that is allocated to each discovery method that is currently running

Parameters	Description
<p>appilog.agent.local.numOfHistoryData = 1</p>	<p>The default number of history results which will be kept for patterns in the Probe Manager repository. Each pattern can set a different number.</p> <p>Note: When task results reach the Probe Manager from the discovery method they are saved in the Probe Manager cache. This value can be changed for specific discovery patterns, by changing the value in the <thresholdInfo numOfHistories="1"> tag in the pattern XML files.</p>
<p>appilog.collectors.minNumberOfObjectsToZip = 1000</p>	<p>If the number of task results that accumulate in the Probe Manager exceeds a certain value, the Probe Manager zips the results before sending them to the Probe Gateway. This parameter determines the minimum number of accumulating task results that is required for zipping.</p>
<p>appilog.agent.Local.MaxNumOfObjectsToZip = 10000</p>	<p>If the number of task results that accumulate in the Probe Manager exceeds a certain value, the Probe Manager zips the results before sending them to the Probe Gateway. This parameter determines the maximum number of accumulating task results that can be zipped.</p>
<p>appilog.agent.local.startP2PDataDistributor = true</p>	<p>If set to true, the P2P listener starts when the Probe Manager begins to work.</p>

Discovery Process Configuration Files

A discovery process needs several parameters to be activated. These parameters specify the method to be used (for example, ping five times before declaring a failure) and on which object the method should be used. If parameters have not been set, the discovery process uses the default parameters defined in the **appilog-remote.properties** file.

In addition, Mercury Application Mapping provides utility files for the operation of the discovery methods. These utility files include standard conversion lists for the discovered data, which convert raw data into information that can be displayed in Mercury Application Mapping, filters that define the scope of the discovery process according to the discovered data type, and so forth. They are stored in the **\Mercury\<Mercury Application Mapping root directory>\root\lib\collectors\discoveryManager\serverData** folder, as follows:

This section contains the following topics:

- “interfaceType.xml” on page 104
- “msDomainNames.xml” on page 104
- “msServerTypes.xml” on page 105
- “oidToHostClass.xml” on page 106
- “OsDescriptionToType.xml” on page 107
- “portNumberToPortName.xml” on page 107
- “syslogFacility.xml” on page 108
- “syslogSeverity.xml” on page 109
- “tcpLinkType.xml” on page 109

interfaceType.xml

Contains a list of known network interface types which convert a discovered interface type's number into a type's name. The interface type's number is discovered using SNMP patterns. Other network interfaces, which are part of the specific discovered network, can be added to the list.

The data is discovered by:

- ▶ SNMP patterns

The parameters are:

- ▶ **interfaceNumberToType number** – the discovered number of a network interface type
- ▶ **string** – the converted name of a discovered interface network type. Under this name, the network interface type appears in the database and in Mercury Application Mapping.

Example:

```
<interfaceNumberToType number="63"  
string="isdn"/>
```

msDomainNames.xml

Contains a list of Microsoft domain types, for defining the scope of the discovery process according to MS domains.

The data is discovered by:

- ▶ MS_NET_Dis_Domain.xml

The parameters are:

- ▶ **MsDomainNames all** – Specify **true** if you want the discovery process to encompass all existing Microsoft domain types, regardless of other definitions that might appear in this file. Specify **false** if you want the discovery process to encompass only the Microsoft domain types that are specified in the file.

- **MsDomainNames** – the Microsoft domain name that is included in the discovery scope.

Example:

```
<MsDomainNames all="false">
<MsDomain>Mercury Application Management Lab</MsDomain>
</MsDomainNames>
```

msServerTypes.xml

Contains a list of all Windows Server types defined in the system (such as, Novell, SQL and NT).

The data is discovered by:

- MS_NET_Dis_Domain.xml

The parameters are:

- **value** – the discovered number of the Server type
- **name** – the converted name of the discovered Server type. The Server type appears under this name in the database and in Mercury Application Mapping.
- **calc** – (calculating) Specify **true** if you want to enter the discovered data into the database (as an attribute of a Host), and continue working with this type of server. Specify **false** if you do not want the discovery methods to continue working with this type of server.

Example:

```
<MsServerType
value="4"
name="SQLSERVER"
calc="true"
/>
```

oidToHostClass.xml

Contains a list of OIDs (a vendor's authoritative identification number assigned to devices in a network for SNMP identification purposes), for all the objects in the system that have an ID. This list is required for mapping objects to their correct class, and for converting the discovered OID number of an operating system or a device into string data.

Note: If an OID is discovered and its details do not appear in the **oidToHostClass.xml** file, its class is registered in the database as **host**.

The data is discovered by:

- SNMP patterns

The parameters are:

- **oidToHostClass class** – the converted class name of the discovered OID. Under this name, the operating system or device appears in the database and in Mercury Application Mapping.
- **vendor** – the vendor of the operating system or device
- **os** – (Optional) a specific operating system's data. For example, a Windows 2000 operating system for the **NT** class.
- **model** – (Optional) a specific model data. For example, an ASX-1200 model of the **atmswitch** class.
- **oid** – the discovered OID

Example:

```
<oidToHostClass class="nt"  
vendor="Microsoft"  
oid="1.3.6.1.4.1.311.1.1.3.1"
```

OsDescriptionToType.xml

Contains a list of descriptions of operating systems which are discovered through the Telnet connection. These descriptions, which are case-sensitive and can be partial, are converted into classes and class types strings. The conversion is needed to determine whether a further query should be performed on the discovered class, and, if so, which kind of query.

The data is discovered by:

- TELNET_NET_Dis_Connection.xml

The parameters are:

- **osDescriptionToType description** – the discovered class description. A class type can include more than one description, so that you can enter a description for each system that refers to this class type.
- **type** – the converted class type of the discovered description
- **class** – the converted class name of the discovered description

Example:

```
<osDescriptionToType description="Microsoft"
type="NT"
class="nt"/>
```

portNumberToPortName.xml

Contains a list of port numbers and their names, for example, HTTP = 80 and firewall = 4000. This list converts a discovered port's number into a port name.

The data is discovered by:

- PSTOOLS_NET_Dis_TCP.xml
- SNMP_NET_Dis_TCP.xml
- TCP_NET_Dis_Port.xml

The parameters are:

- ▶ **portinfo portProtocol** – the protocol used for the port number discovery
- ▶ **portNumber** – the discovered port name
- ▶ **portName** – the discovered port number is converted into its name
- ▶ **discover** – the port to be queried
- ▶ **tcpDiscover** – the port to be queried using TCP connections

Example:

```
<portInfo portProtocol="tcp"  
portNumber="21"  
portName="ftp"  
discover="1">  
tcpDiscover="1"/>
```

syslogFacility.xml

Contains a list that translates types of event numbers that were sent from SysLog into specific strings that appear in Mercury Application Mapping event messages.

The data is discovered by:

- ▶ SYSLOG_NET_Lis_Cisco.xml
- ▶ SYSLOG_NET_Lis_SnifferTcp.xml
- ▶ SYSLOG_NET_Lis_SnifferUdp.xml

The parameters are:

- ▶ **syslogFacilityNumberToName number** – the discovered event number
- ▶ **string** – the converted event message. The discovered event appears in the database and in Mercury Application Mapping with this message.

Example:

```
<syslogFacilityNumberToName number="2"  
string="mail system"/>
```


syslogSeverity.xml

Translates discovered severity numbers of SysLog events into Mercury Application Mapping textual and numerical event severities.

The data is discovered by:

- SYSLOG_NET_Lis_Cisco.xml
- SYSLOG_NET_Lis_SnifferTcp.xml
- SYSLOG_NET_Lis_SnifferUdp.xml

The parameters are:

- **syslogSeverityNumberToName number** – the discovered SysLog event severity number
- **string** – the converted string that indicates the severity of the discovered event
- **state** – the converted number that indicates the severity of the discovered event

Example:

```
<syslogSeverityNumberToName number="2"
string="Critical"
state="7"/>
```

tcpLinkType.xml

Contains a list of all TCP link types in the system that translate TCP link type numbers into strings indicating the link state. In addition, a definition is set for each TCP link which determines whether the discovery methods are required to further query this type of link.

The data is discovered by:

- PSTOOLS_NET_Dis_TCP.xml
- SNMP_NET_Dis_TCP.xml

The definitions are:

- ▶ **tcpLinkType number** – the discovered TCP link type number
- ▶ **type** – the converted string that indicates the state of the TCP link type
- ▶ **discover** – whether to query the discovered TCP link type. Yes = **1**, No = **0**

Example:

```
<tcpLinkType number="2"  
type="listen"  
discover="1"/>
```

The Discovery System Directory Structure

The folders and files of the Mercury Application Mapping discovery system are located in: **Mercury\<Mercury Application Mapping root directory>\root\lib\collectors**.

The following sections describe the contents and use of these folders and files for each of the Mercury Application Mapping discovery components. For details on the Mercury Application Mapping discovery components, see the *Mercury Application Mapping Discovery Process Tutorial*.

Note: When Mercury Application Mapping discovery components are installed on different workstations, some of the files and folder are copied and stored on more than one workstation to provide the data locally for each component.

This section contains the following topics:

- “Main Folder” on page 111
- “The discoveryManager Folder” on page 111
- “The p2pViewer Folder” on page 112
- “The patches Folder” on page 112
- “The probeGateway Folder” on page 112
- “The probeManager Folder” on page 113
- “The remoteAgent Folder” on page 114

Main Folder

- **appilog-remote.properties** – the Probe Manager and Probe Gateway configuration file (for details, see “The Probe Gateway and Probe Manager Configuration File” on page 91).
- **delCollectors.bat** – a command file that deletes the discovery tasks data from the Probe Gateway and Probe Manager, including all tables, repositories, and pending tasks, and builds the data from scratch
- **jms.properties** – a configuration file for RMI communication mode
- **jndi.properties** – a configuration file that contains the connection data of the Mercury Application Mapping server, such as IP address, user name, and password. The Probe Gateway uses this data when contacting the Server in RMI mode.
- **versions.properties** – contains the version and build details of the installed collectors

The discoveryManager Folder

- **patternToClass.xml** – contains the definitions of classes whose performance is monitored and definitions of the monitoring patterns.
- **PQL - PROBES IN DB.xml** – Used by the infrastructure for the discovery process. This file should not be changed in any way.

The **discoveryManager** folder includes the following:

- **The patterns Folder** – Mercury Application Mapping discovery patterns are XML files used by the discovery system. This folder is divided into the following subfolders:
 - **all** – the list of Mercury Application Mapping discovery patterns
 - **pm** – the list of discovery patterns that can monitor class performance.
 - **templates** – for internal use
 - **test** – for internal use
- **Patterns.xsd** – defines the XML schema of the discovery pattern
- **PatternTestHelper.xsd** – defines the XML schema of the discovery pattern tester

The p2pViewer Folder

- **mam4j-viewer.properties** – for internal use
- **p2pConfig.xml** – for internal use
- **P2PFilters.xml** – for internal use
- **P2PSettings.properties** – for internal use

This folder is divided into the following subfolders:

- **icons** – for internal use
- **recordings** – contains p2p viewer recordings that were saved

The patches Folder

Contains patches for upgrading Mercury Application Mapping

The probeGateway Folder

- **localsDataForProbe.xml** – contains the IP addresses of the Probe Managers of each Probe Gateway
- **mam4j-probe.properties** – the configuration file of the Probe Gateway log

- ▶ **probeTables.xml** – contains a definition set of the Probe Gateway repository tables

The **probeGateway** subfolder includes the following folders:

- ▶ **serverData** – a copy of the serverData folder needed for the operation of the Probe Gateway (for details, see “Discovery Process Configuration Management” on page 90).
- ▶ **servicesJars** – for internal use
- ▶ **userExt** – for internal use

The probeManager Folder

- ▶ **mam4j-local.properties** – the configuration file of the Probe Manager log

The **probeManager** subfolder includes the following:

- ▶ **The databases Folder** – Includes the Probe Gateway repository. Task requests and task results that are not delivered to the Probe Manager or to the Server due to connection problems are saved to this repository. Once the requests and results are delivered to their destination, they are deleted from the repository.
- ▶ **The localRepository Folder** – The Probe Manager main repository. Task results are saved to this repository. For each task invocation, the last results are saved. If additional past results need to be saved (as specified in the **<thresholdInfo numOfHistories="1">** tag in the pattern’s task), other results are also saved to this repository.
- ▶ **The netlinks Folder** – for internal use
- ▶ **The p2p Folder** – for internal use
- ▶ **The scripts Folder** – for internal use
- ▶ **The serverData Folder** – Includes editable files that contain definitions and parameters that mark out the discovery scope and provide necessary data, such as passwords, needed for the operation of the discovery process (for details, see “Discovery Process Configuration Management” on page 90).
- ▶ **The servicesJars Folder** – for internal use

- ▶ **The tasks Folder** – The Probe Manager repository which contains a list of discovery tasks that run on the Probe Manager. These tasks have been allocated to the specific Probe Manager which has an unlimited running schedule. That is, the tasks are not limited by the number of times they are meant to run or by any time table. The purpose of this repository is to prevent the loss of task data in case of a crash.
- ▶ **tmp** – for internal use
- ▶ **The unsentTaskResults Folder** – for internal use
- ▶ **userExt** – for internal use

The remoteAgent Folder

- ▶ **appilog-agent-configuration.properties** – for internal use
- ▶ **mam4j-agent.properties** – for internal use

5

User Interface Settings Configuration

This chapter explains how to configure the targets, patterns, parameters, and graph elements of the Performance Monitoring feature and how to use the **gui.properties** file to configure Mercury Application Mapping..

This chapter describes:	On page:
Configuring the User Interface Properties	116
Adding Customized Icons to Mercury Application Mapping	117

Configuring the User Interface Properties

You configure user interface properties with the **gui.properties** file located in `\Mercury\<Mercury Application Mapping root directory>\root\lib\web`. You can change the following attributes:

- ▶ **appilog.gui.PollingDelay=30000** – the time interval (in milliseconds) between each request for updates from the server
- ▶ **appilog.gui.MaxConnectedTries=10** – the number of times Mercury Application Mapping contacts the Server, if a request for updates fails
- ▶ **appilog.gui.MaxObjectCountGuiForTS=900** – the maximum number of objects in one layer that can be displayed in Map format

When the layer contains more objects than indicated here, Mercury Application Mapping displays them in table format.

- ▶ **appilog.gui.MaxObjectCountGui=1300** – the maximum number of objects in one layer that can be displayed in table format (Details window)

When the layer contains more objects than indicated here, Mercury Application Mapping will not open the layer at all.

Note: The **MaxObjectCountGuiForTS** and **MaxObjectCountGui** parameters also appear in the **appilogConfig.properties** file. If you change a parameter in the **gui.properties** file, you must also change it in the **appilogConfig.properties** file.

Adding Customized Icons to Mercury Application Mapping

Objects that are displayed in the Map View, comprise the following components:



- A body icon
- A family icon – a group to which the body icon belongs, for example, Network or SAP, that appears in the upper left-hand corner of the object.

Following is an object as it appears in the Mercury Application Mapping Map View:



For more information on object icons, see, “Mercury Application Mapping Quick Tour” in the *Mercury Application Mapping User’s Guide*.

Mercury Application Mapping enables you to create your own customized icons. The icons you create appear in the **Class Family** and **Class Main Icon** lists in the Class wizard where you can assign the icons to a class. For details, see “Assigning an Icon to a Class” in the *Mercury Application Mapping User’s Guide*.

This section includes the following topics:

- “Icon Name Format” on page 117
- “Creating a Customized Body Icon” on page 118
- “Creating a Customized Family Icon” on page 119

Icon Name Format

Each icon is kept in a different folder and must have one of the following formats:

`<icon name>_<icon color>_<icon size>`

`<icon name>_<icon color>_<parent_object>_<icon size>`

where

- `<icon name>` is the name of the icon.
- `<icon_color>` is the color that indicates the object’s status level.

- ▶ **<parent_object>** indicates that there is an additional object layer(s) beneath it.
- ▶ **<icon_size>** is the size of the icon measured in pixels. There are two sizes: 16 pixels and 32 pixels. The 16 pixel icons appear in the Explorer pane and the 32 pixel icons appear in the Map View.

Creating a Customized Body Icon

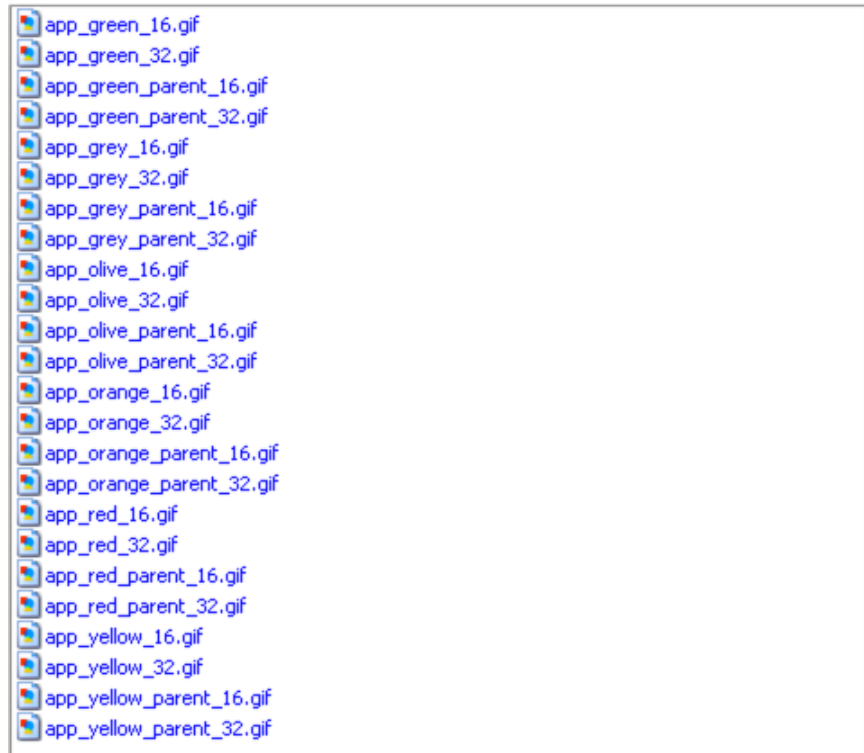
Body icons must be placed in the following location:

\Mercury\<Mercury Application Mapping root directory>\root\lib\gui\images\classes\icons\body

To create a customized body icon:

Create a new subfolder in the body icon directory specified above and add the new icon to it. The name of the icon must be identical to the name of the subfolder in which it is kept and is case-sensitive. For example, if the name of your icon is **app**, then the name of the subfolder must also be **app**, and not **App** or **APP**. There must be 24 instances of each body icon in the subfolder.

For example:



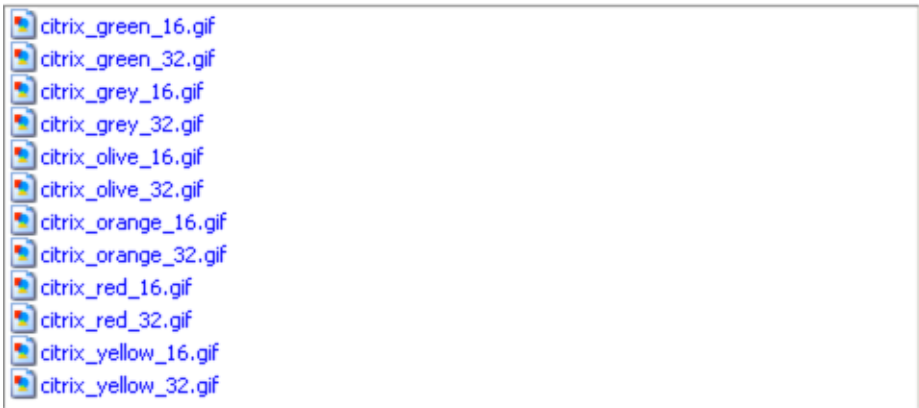
Creating a Customized Family Icon

Family icons must be placed in the following location:
\Mercury\<<Mercury Application Mapping root directory>\root\lib\gui\images\classes\icons\family

To create a customized family icon:

Create a new subfolder in the family icon directory specified above and add the new icon to it. The name of the icon must be identical to the name of the subfolder in which it is kept and is case-sensitive. For example, if the name of your icon is citrix, then the name of the subfolder must also be citrix, and not Citrix or CITRIX. There must be 12 instances of each family icon in the subfolder.

For example:



6

Server Properties Configuration

This chapter describes the configuration parameters required by the Server for each of its subsystems, how to start and shut down the server, how to control the log output and to customize log parameters, and how to define the list of Servers in the Login page..

This chapter describes:	On page:
Configuring the Server Subsystem Parameters	122
Starting and Shutting Down the Mercury Application Mapping Server	136
Mercury Application Mapping Log Files and Directories	137
Defining Log Properties	139
Reducing Log Load	141
Defining Server Login Data	141

Configuring the Server Subsystem Parameters

The **appilogConfig.properties** file contains the general configuration parameters required by the Server for each of its subsystems.

This section contains the following topics:

- “Configuring the Server Subsystem Parameters” on page 122
- “Event System” on page 123
- “Collectors” on page 123
- “TQL” on page 124
- “Map Server” on page 129
- “Proxy” on page 132
- “Report System” on page 132
- “WorldManager System” on page 133
- “General Stuff properties” on page 134
- “Export Database Properties” on page 135
- “Gui Resource Path” on page 136
- “ACL Properties” on page 136

Configuring the Server Subsystem Parameters

You can configure the server subsystem parameters.

To configure the server subsystem parameters:

- 1** Open **appilogConfig.properties** in a text editor. The file is located in:
Mercury\<**Mercury Application Mapping root directory**>\root\lib\server.
- 2** Edit the server subsystem parameters as described in the following sections.
- 3** Save the file.

Event System

- **appilog.event.topicConnectionFactory.ClassName=appilog.TopicConnectionFactory** – the location of the JMS subsystem on the server, needed for establishing a connection with the JMS subsystem

Collectors

- **appilog.collectors.Dir=Mercury/<Mercury Application Mapping root directory>/root/lib/collectors/** – The location of the discovery system files. This parameter is entered automatically during installation.
- **appilog.collectors.RetryTimes=10** – The number of times the server tries to save incoming task results in the database.
- **appilog.collectors.RetryDelaySeconds=60** – The number of seconds between each attempt to save task results to the database (as described above in **appilog.collectors.RetryTimes=10**).
- **appilog.collectors.MaxTasksPerJmsMessage=50** – The number of task requests that are sent together in one package from the server to the Probe Gateway.

When configuring this parameter, take into consideration system connection capabilities and performance.

- **appilog.collectors.NotDeleteLimitedTasks=true** – Defines whether to automatically delete limited tasks once their trigger TQL is no longer realized.

Limited tasks are tasks that have limited activation schedules, that is, the number of times they are re-activated is pre-defined.

true – limited tasks are not actively deleted, since their limited scheduler causes them to expire.

- **appilog.collectors.DefaultDomain=niceDomain** – The domain that defines which Probe Gateway receives task requests for objects that have unknown domains, when they are defined with the **enforce-dispatch** tag.
- **appilog.collectors.domainSearchAttributes=ip-domain;ip-address;application_ip;snmp_ip** – The attributes to be used in the discovery domain selection process.

TQL

- ▶ The following are parameters that enable you to prevent large TQL queries from over-consuming server memory. Once these parameters are exceeded, Mercury Application Mapping either issues warning to the **pqlfuse.log** file or uses TQL circuit breakers to stop the calculation of large TQLs and deactivates them.
 - ▶ **appilog.pql.fuse.error=1000000** – the upper threshold for the number of objects being calculated in all TQLs currently running. Exceeding this number causes Mercury Application Mapping to start deactivating the TQLs that are currently running in the system.
 - ▶ **appilog.pql.fuse.warning=650000** – the lower threshold for the number of objects being calculated in all TQLs currently running. Exceeding this number issues a warning in the **pqlfuse.log** file.
 - ▶ **appilog.pql.fuse.inActiveCount=10** – the number of TQLs that are deactivated if the upper threshold (**appilog.pql.fuse.error**) is exceeded. Mercury Application Mapping deactivates the ten largest TQLs (the TQLs with the largest number of objects), starting with the largest one.

Note: To activate the TQLs once again, you need to change the TQL definition and save the new definition. For details, see “Defining Topology Query Language (TQL) Queries” in the *Mercury Application Mapping User’s Guide*.

- ▶ **appilog.pql.fuse.calc.error=50000** – the number of results that are calculated per single TQL when the system is running in error mode (meaning that TQLs are being deactivated)
- ▶ **appilog.pql.fuse.calc.normal=600000** – the number of results that are calculated per single TQL when the system is running under the threshold of **appilog.pql.fuse.warning**.
- ▶ **appilog.pql.fuse.calc.warning=180000** – the number of results that are calculated per single TQL when the system is running between the limitations defined in **appilog.pql.fuse.warning** and **appilog.pql.fuse.error**.

- **appilog.pql.fuse.group=2,3** – the types of TQLs you want Mercury Application Mapping to deactivate (**2** = Discovery TQLs, **3** = View TQLs)

Note: The defaults settings are **2** and **3**. It is recommended not to change the default settings.

The following is a sample message that appears in the **pqlfuse.log** file when there is a change in state from Normal to Warning or from Warning to Error:

```
2005-11-15 16:19:50,650> INFO - PQL Fuse created: warning=40000 error=58300
[ExecuteThread: '11' for queue: 'weblogic.kernel.Default']
<2005-11-15 16:19:50,650> INFO - Calculation thresholds: normal=70000,
warning=60000, error=50000 [ExecuteThread: '11' for queue: 'weblogic.kernel.Default']
<2005-11-15 16:19:54,071> WARN - state changed from Normal to Warning count
is:58363 [Thread-11]
<2005-11-15 16:19:54,071> INFO - current state is: Warning[58363] [Thread-11]
<2005-11-15 16:19:59,071> WARN - state changed from Warning to Error count is:58363
[Thread-11]
<2005-11-15 16:19:59,071> WARN - going to disable top:2 [Thread-11]
<2005-11-15 16:19:59,071> WARN - going to disable pql: MQ_All_Objects [Thread-11]
<2005-11-15 16:19:59,071> WARN - going to disable pql: MQ_Network_Objects
[Thread-11]
<2005-11-15 16:19:59,071> INFO - current state is: Error[58363] [Thread-11]
<2005-11-15 16:20:04,071> INFO - state changed from Error to Warning count is:3446
[Thread-11]
<2005-11-15 16:20:04,071> INFO - current state is: Warning[3446] [Thread-11]
<2005-11-15 16:20:09,071> INFO - state changed from Warning to Normal count is:3446
[Thread-11]
<2005-11-15 16:20:09,071> INFO - current state is: Normal[3446] [Thread-11]
<2005-11-15 16:20:14,071> INFO - current state is: Normal[3446] [Thread-11]
```

The following is a sample message that appears in the **pqlfuse.log** file when the TQL size exceeds the defined size for a specific state.

```
<2005-11-15 16:27:51,025> ERROR - failed to calc pql size during calc exceeded the
maximum size [72004>70000] [PQL_MQ_All_Objects #1]
<2005-11-15 16:27:51,025> ERROR - failed to calculate TQL [PQL_MQ_All_Objects #1]
[ERROR - 20006] appilog.server.world.pql.fuse.PqlCalcSizeException: failed to calc pql
size during calc exceeded the maximum size [72004>70000]
at appilog.server.world.pql.fuse.AbstractPqlFuse$SimpleCalcFuse.testCalc
(AbstractPqlFuse.java:79)
    at appilog.server.world.pql.fuse.PqlFuse.testGraphCalc(PqlFuse.java:45)
    at appilog.server.world.pql.PQLGraph.testCalc(PQLGraph.java:2010)
    at appilog.server.world.pql.PQLGraph.calculateNotReqInnerLinks(PQLGraph.java:902)
    at appilog.server.world.pql.PQLGraph.calculateInnerLinks(PQLGraph.java:796)
    at appilog.server.world.pql.PQLGraphCalculator.calculate
(PQLGraphCalculator.java:79)
    at appilog.server.world.pql.Pattern.getPQLResults(Pattern.java:1002)
    at appilog.server.world.pql.Pattern.recalculate(Pattern.java:794)
    at appilog.server.world.pql.Pattern.calculate(Pattern.java:575)
    at appilog.server.world.pql.beans.pqlcalculator.PQLCalculatorMessageBean.calculate
(PQLCalculatorMessageBean.java:59)
    at appilog.server.world.pql.beans.pqlcalculator.PQLCalculatorMessageBean.
onEventMessage
(PQLCalculatorMessageBean.java:40)
    at appilog.server.model.beans.AppilogMessageModel.onMessage
(AppilogMessageModel.java:80)
    at weblogic.ejb20.internal.MDListener.execute(MDListener.java:382)
    at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:197)
    at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:170)
```

- **appilog.pql.PrintStatisticInterval=60** – the interval (in minutes) between each printing of a TQL statistics in the **pql_statistics.log** file, located in:
\Mercury\<<Mercury Application Mapping root directory>\root\logs
- **appilog.pql.SaveResultsInterval=10** – the interval (in minutes) between each saving of TQL calculation results in the database

This value does not need to be low because even if the server crashes, results are recalculated upon restarting and are not lost.
- **appilog.pql.MaxEventsInc=1000** – the maximum number of events related to a TQL, above which the TQL is not calculated incrementally, but is rebuilt
- **appilog.pql.MaxPatterns=200** – the maximum number of TQLs in the system

The following definitions determine when a TQL with a certain priority level is calculated. There are six definitions for each of the four priority levels: Low, Medium, High, and Express.

Notes:

- ▶ The TQL priority level is set in the TQL Properties dialog box in the TQL Builder (for details, see Chapter 4, “Defining Topology Query Language (TQL) Queries,” in the *Mercury Application Mapping User’s Guide*).
 - ▶ All the following definitions are measured in milliseconds.
 - ▶ For a minimum definition to come into effect, the minimum definitions need to be fulfilled. (Minimum definitions are connected by the AND operator.)
 - ▶ For a maximum definition to come into effect, one maximum definition must be fulfilled. (Maximum definitions are connected by the OR operator.)
-

Priority Level	Definitions
The minimum time that has passed since the last TQL calculation.	appilog.pql.lowPriority.MinEventsTimeout=600000 appilog.pql.medPriority.MinEventsTimeout=60000 appilog.pql.highPriority.MinEventsTimeout=60000 appilog.pql.expressPriority.MinEventsTimeout=5000
The maximum time that has passed since the last TQL calculation.	appilog.pql.lowPriority.MaxEventsTimeout=900000 appilog.pql.medPriority.MaxEventsTimeout=540000 appilog.pql.highPriority.MaxEventsTimeout=180000 appilog.pql.expressPriority.MaxEventsTimeout=180000
The minimum number of pending JMS event messages.	appilog.pql.lowPriority.MinEventsNumber=5 appilog.pql.medPriority.MinEventsNumber=5 appilog.pql.highPriority.MinEventsNumber=1 appilog.pql.expressPriority.MinEventsNumber=1

Priority Level	Definitions
The maximum number of pending JMS event messages.	appilog.pql.lowPriority.MaxEventsNumber=1000 appilog.pql.medPriority.MaxEventsNumber=1000 appilog.pql.highPriority.MaxEventsNumber=1000 appilog.pql.expressPriority.MaxEventsNumber=10
The minimum time that has passed since the arrival of the last JMS event message.	appilog.pql.lowPriority.MinEventsFreq=60000 appilog.pql.medPriority.MinEventsFreq=60000 appilog.pql.highPriority.MinEventsFreq=30000 appilog.pql.expressPriority.MinEventsFreq=5000
The maximum time that has passed since the arrival of the last JMS event message.	appilog.pql.lowPriority.MaxEventsFreq=180000 appilog.pql.medPriority.MaxEventsFreq=180000 appilog.pql.highPriority.MaxEventsFreq=180000 appilog.pql.expressPriority.MaxEventsFreq=30000

Map Server

The following definitions determine the calculation and display of objects and views:

- **appilog.map.WebCachCleanTimeOut=120000** – The WebCach timeout in milliseconds. Default is 20 minutes.
- **appilog.map.slo.beginOfDay = 08:00:00** – The hour from which availability is calculated.
- **appilog.map.slo.beginOfWeek = 2** – The day in the week from which availability is calculated. Sunday = **1**, Monday = **2**, and so forth.
- **appilog.map.slo.beginOfMonth = 1** – The date in the month from which availability is calculated.

The system calculates availability according to the default values as follows: a day begins at 8:00 AM, a week begins on a Monday, and a month begins on the first of the month.

The following definitions specify the intervals between each calculation in each of the calculation cycles (day, week, and month):

- **appilog.map.slo.interval.units=hour** – The time unit of the calculation interval: minute, hour, day, or week.
- **appilog.map.slo.interval.basic=1** – The number of time units that serves as the basic interval unit for the calculation.
- **appilog.map.slo.interval.day=2** – The number of basic interval units for an availability calculation during a day.
- **appilog.map.slo.interval.week=10** – The number of basic interval units for an availability calculation during a week.
- **appilog.map.slo.interval.month=24** – The number of basic interval units for an availability calculation during a month.

The system calculates intervals according to the default values as follows: if the basic unit is one hour, intervals are calculated every two hours for a day, every 10 hours for a week, and every 24 hours for a month.

- **appilog.map.MaxViews=102** – The maximum number of views that can be created in the system.

When the number of existing views reaches the maximum value, you cannot create additional views, but the existing views are not deleted. To add new views, you must manually delete some of the existing views or increase this value.

The following definitions determine the minimum requirements for performing automatic rebuild of views. Both definitions must be fulfilled for automatic rebuilding to be activated.

- **appilog.map.MinRebuild =100** – The minimum number of accumulated JMS event messages in a queue required for performing an automatic rebuild.
- **appilog.map.MoreThanRebuild =0.25** – This definition, like **appilog.map.MinRebuild**, specifies a minimum number of accumulated event messages for performing an automatic rebuild.

However, in this definition the number is based on the percentage of accumulated event message from the total number of objects in the view. The default value is 0.25, that is, when the number of accumulated event messages is more than a quarter of the view's object number, an automatic rebuild will be performed.

The following definitions determine the display options of a large number of objects:

- ▶ **appilog.map.MaxObjectCountGui=1300** – The maximum number of objects in one layer that can be displayed in table format.
- ▶ **appilog.map.MaxObjectCountGuiForTS=900** – The maximum number of objects in one layer that can be displayed in map format.

When a layer contains more objects than indicated here, it displays them in table format.

- ▶ **appilog.map.MaxLabelSizeForLayer=15** – The maximum number of characters that can be in an object’s label.
- ▶ **appilog.map.LayerBulkSize = 1000** – The number of objects that are displayed in each bulk in the Details window (for details on the Details window, see “Viewing Layer Details”, in *Mercury Application Mapping User’s Guide*).
- ▶ **appilog.map.SymbolBulkSize = 1000** – The maximum number of updated objects in each bulk that reaches the Mercury Application Mapping user interface.
- ▶ **appilog.map.EventBulkSize = 1000** – The maximum number of events in each bulk that reaches the Mercury Application Mapping user interface.
- ▶ **appilog.map.MaxServletResponseTime = 10000** – For debug mode only. – The maximum server response time for a user request. The default is 10000 and is measured in milliseconds. If this time is exceeded, a message is written to **web.log**, which is located at: **\Mercury\<<Mercury Application Mapping root directory>\root\logs**.
- ▶ **appilog.map.getLayerUnderLinkExtended = true** – Some links represent several links between objects that exist on different layers. Set this parameter to **true** to display all the links that are related to the link from all layers. Set this parameter to **false** to display all the links that are related to the link just in the selected view.
- ▶ **appilog.map.maxServerQueueSize = 5000** – The maximum number of accumulated changes that can be stored in a queue in the Mercury Application Mapping server before being dispatched to the Mercury Application Mapping user interface.

Proxy

This section contains definitions for proxy server parameters as well as the default server identification.

- **appilog.proxy.maxThreads = 100** – The maximum number of servers the proxy can connect to simultaneously.
- **appilog.proxy.connectionTimeout= 60000** – The maximum time the proxy waits for results from the servers. If the proxy does not receive results from one server, it will turn to another server to get results. This definition applies only to cases where the server did not respond due to network problems. Connection timeout is measured in milliseconds.
- **appilog.proxy.resultTimeOut = 60000** – the maximum time the proxy waits for results from the servers. If the proxy does not receive results from one server, it will turn to another server to get results. This definition applies only to cases where the reason the server did not respond is **NOT** due to network problems. Connection timeout is measured in milliseconds.
- **appilog.serverid = 1** – The default server identification. The server that sends results to the Mercury Application Mapping user interface identifies itself as the server from which the results were sent. If the server cannot find its own IP address, it uses the default server identification.

Report System

The following definitions determine several parameters of the display and calculation of System Reports. The first definition applies to Map and Host Reports as well.

- **appilog.report.NoDataMsg=There is no data for this report.** – The text that appears in a report cell when no data is available.
- **appilog.report.IdColName=^^OBJECT_ID^^** – For internal use.
- **appilog.report.DebugMode=false** – Whether to save the temporary tables that are created for the report calculation to the database.

The default value is **false**, meaning the tables are not saved to the database after report creation.

- **appilog.report.LastAvailabilityDurationUnits=day** – The time unit of the **availabilitylast** function.

The available options are: second, minute, hour, day. This function can be used when defining system reports in the Report Manager. It appears on the function list in the Column Definition Wizard, accessed from the Report Node Definition dialog box.

- **appilog.report.Directory=C:/Mercury/<Mercury Application Mapping root directory>/root/lib/server/reports/** – Currently not in use.
- **appilog.report.CsvDelimiter=,** – The character that separates fields when you create reports in Excel format.
- **appilog.report.maxRows=2000** – Maximum number of lines in the report.
- **appilog.report.maxSize=80000** – Represents the number of columns multiplied by the number of rows.

WorldManager System

The following Server timeout definitions determine when and for how long the Server pauses:

- **appilog.worlmanager.Timeout=10** – The time (in milliseconds) of the server pause. The default is 10 milliseconds.
- **appilog.worlmanager.ObjectCount=1000** – the number of processed objects after which the Server pauses. The default is 1000 objects.

The default definitions determine that the Server pauses for one second after 1000 processed objects.

- **appilog.worlmanager.ContainerCacheSize=10000** – the maximum number of containers in a cache that maps the unique ID number (UID) of an object to its ID number.
- **appilog.worlmanager.ClassCacheSize=1000000** – the maximum number of classes in a cache that maps the objects ID number to its class
- **appilog.worlmanager.LabelFunctionCacheSize=10000** – the maximum number of calculations that map an object ID to its label in a cache
- **appilog.worlmanager.RawEventRemoveChunkSize=50000** – the maximum number of raw events that can be deleted from the database at one time

- **appilog.wolrdmanager.ObjectRemoveChunkSize=1000** – the maximum number of objects that can be deleted from the database at one time

General Stuff properties

- **appilog.general.TimeZoneOffset=0** – determines whether the system time zone should be adjusted to summer time

If your system is installed in the USA, leave the default value (0), since the time zone is updated automatically. If your system is installed in Israel, change the value to “1”.

- **appilog.general.Log4jPropFile=mam4j-non-debug.properties** – determines the name of the server log file (for details on system log files, see “Defining Log Properties” on page 139).

- **appilog.general.MessageDispatcherTimeout=10000** – determines the maximum time the JMS tries to send events to the event system. Dispatch timeout is measured in milliseconds.

- **appilog.general.ActiveEventHashTableLimit=10000** – determines the minimum number of accumulated active events, after which a message is sent to the WebLogic console, indicating the number of active events in the system

When an additional 1000 active events accumulate, a new message is sent. This number can assist you in keeping track of the active events accumulating in the system memory.

- **appilog.general.ContainerCacheSize=50000** – The number of objects that are contained in the system cache

When this number is exceeded, the first counted objects are deleted from the cache. The default number is 10000 objects.

- **appilog.general.ActiveEventBulkSize=10000** – the default amount of active events that are automatically saved to the database

- **appilog.general.ActiveEventBulkTime=60** – the default time interval (in minutes) that determines, together with the **ActiveEventBulkSize** definition, the activation of an automatic saving of active events

The default time is 60 minutes, that is, active events in the system are saved automatically once an hour (for details on these definitions, see “Configuring the Automatic Saving of Active Events” on page 43).

- **appilog.general.MailServer=mail.appilog.com** – the address of the mail server from which e-mails and notifications are sent
- **appilog.general.MailFrom=admin@mercury.com** – the address from which e-mails are sent
- **appilog.general.TrapPort=162** – Obsolete.
- **appilog.general.EnterpriseOid=1.1.1** – Obsolete.
- **appilog.general.LogAction=true** – determines whether user actions performed in Mercury Application Mapping create raw events
true means that a raw event is created when the user performs an action.
- **appilog.general.RulesPackageName=Rules.zip** – For internal use only. Do not edit in any way.
- **appilog.general.FilterCorrelation=true** – Determines whether or not to send correlation event to the user interface. If you set this parameter to True, correlation events will NOT be sent to the user interface.
- **appilog.general.WorldTopicLimit=5000** – determines the maximum number of JMS event messages that can be accumulated in a JMS queue
When the number of pending event messages has reached this limit, new event messages are kept at the Probe Gateway until the messages on the queue are handled.
- **appilog.general.DurationUnit=minute** – the time unit of the Duration operator used in Mercury Application Mapping
The available options are second, minute, hour, and day.
- **appilog.general.XmlDefaultEncoding=ISO-8859-1** – For internal use only. Do not edit in any way.
- **appilog.general.RootDir=C:/Mercury/<Mercury Application Mapping root directory>/root** – For internal use only. Do not edit in any way.

Export Database Properties

The definitions that appear in this section determine the parameters of the automatic export scheduler. They are described in “Exporting the Topology Database Automatically” on page 145.

Gui Resource Path

- **appilog.general.guiResourcePath=C:/Mercury/<Mercury Application Mapping root directory>/root/lib/gui** – For internal use only. Do not edit in any way.

ACL Properties

- **appilog.acl.administrator=Administrators role** – the definition of a role with Administrator access rights that has unrestricted access to all Mercury Application Mapping functions.

Starting and Shutting Down the Mercury Application Mapping Server

You can operate Mercury Application Mapping Server server either as a Windows service or as an application with an open console.

This section contains the following topics:

- “Starting the Mercury Application Mapping Server” on page 136
- “Shutting Down the Mercury Application Mapping Server” on page 137

Starting the Mercury Application Mapping Server

You can start the Mercury Application Mapping Server as an application or as a service.

To start the Mercury Application Mapping Server as an application:

- Double-click the **Start WebLogic Server** shortcut created upon installation.

To start the Mercury Application Mapping Server as a service:

- 1 Open the Windows Services page.
- 2 From the **Name** list, select the relevant server and click the **Start Service** button on the toolbar.

Shutting Down the Mercury Application Mapping Server

When you shut down the Server, the Server saves the data that is stored in its cache (for example, active events) to the database and updates its current state. This update prevents recurring and unnecessary calculations upon restart.

To shut down the Mercury Application Mapping Server:

- 1** Double-click the `server_shutdown.cmd` file located in:
`\Mercury\<<Mercury Application Mapping root directory>\scripts`
- 2** Answer **Y** to the question in the Command Prompt window and press **Enter**.

To shut down the Mercury Application Mapping Server service:

Open the Windows Services page. Select the relevant Server in the **Name** list and click the **Stop Service** button on the toolbar.

Mercury Application Mapping Log Files and Directories

Mercury Application Mapping, Mercury Application Mapping Server, Probe Gateway, and Probe Manager output logs either to specific log files or to the component's console that receives and collects the log events that appear in the console log. The console log for each Mercury Application Mapping component can also be configured, as described in "Defining Log Properties" on page 139.

The log files are located in:

`\Mercury\<<Mercury Application Mapping root directory>\root\logs`

- **probemanager-infra.log** – contains activated discovery patterns and tasks that are passed to the Probe Manager
- **collectorsManagement.log** – contains information about the discovery process on the Mercury Application Mapping server side
- **events.log** – contains technical information about events that occur in the Mercury Application Mapping system
- **mam_gui.log** – contains error messages, warnings, and information from the Mercury Application Mapping user interface

- **mam_gui_errors.log** – contains only error messages from the Mercury Application Mapping user interface
- **mapserver.log** – contains information on occurrences in the Map View tab, such as objects that have been deleted from a view
- **packaging_info.log** – displays information on deployed packages
- **pending.log** – contains the number of pending messages in the JMS queue
- **pql.log** – contains general information about TQL errors
- **pql_statistics.log** – contains the start and end of TQL calculations as well as statistics about TQL calculations
- **pql directory** – contains log files with information about each TQL in the system
- **probemanager-infra.log** – contains data about the infrastructure of the Probe Manager (that is not connected to a specific service)
- **probemanager-services.log** – contains the discovery tasks being passed to the Probe Manager and the results of the discovery task execution
- **tasks.log** – contains all the discovery tasks that were sent to the Probe Gateway
- **web.log** – contains all the requests from the Mercury Application Mapping user interface
- **world.log** – contains new objects that were discovered and inserted into the database.
- **world_statistics.log** – contains statistical information about new objects that were discovered and inserted into the database.
- **wrapperLocal.log** – contains error messages and other information received from the Probe Manager
- **wrapperProbe.log** – contains error messages and other information received from the Probe Gateway
- **wrapperServer.log** – contains error messages and other information received from the Mercury Application Mapping server
- **xsl-filter.log** – contains data regarding the xsl transformation of the data for Mercury Application Mapping WebView

Defining Log Properties

Log files are based on a named log hierarchy that enables you to control the level of log statements that are sent to the log files. There are five output levels: DEBUG, INFO, WARNING, ERROR, and FATAL.

In addition to controlling the log output level, you can also customize other parameters, such as the format of the log output and the log target. Each Mercury Application Mapping component includes a configuration file that contains definitions, as follows:

Component	Name of configuration file	Location
Mercury Application Mapping	mam4j_gui.properties	Mercury\<<Mercury Application Mapping root directory>\root\lib\gui
Mercury Application Mapping Server	mam4j.properties	Mercury\<<Mercury Application Mapping root directory>\root\lib\server
Probe Gateway	mam4j-probe.properties	\Mercury\<<Mercury Application Mapping root directory>\root\lib\collectors\probeGateway
Probe Manager	mam4j-local.properties	\Mercury\<<Mercury Application Mapping root directory>\root\lib\collectors\probeManager

To set log properties:

- 1** Open the log file you want to edit with a text editor.
- 2** Make the changes and save the file.
- 3** Restart the Mercury Application Mapping Server.

Mercury Application Mapping uses the Log4J system of logging (for details, see the Log4j Project documentation at <http://jakarta.apache.org/log4j/docs/>).

For each log file, the following definitions are set in each of the configuration files:

- **log4j.category.appilog.server.events=DEBUG, EVENTS** – the name of the file that is scanned for errors
- **log4j.appender.EVENTS=org.apache.log4j.RollingFileAppender** – a file, console, mail, event message, and so forth
- **log4j.appender.EVENTS.File=Mercury/<Mercury Application Mapping root directory>/root/logs/events.log** – the name and path of the log file
- **log4j.appender.EVENTS.Append=false** – determines whether to create a new log file each time the Server restarts (false), or to continue printing the output in the existing log file (true)
- **log4j.appender.EVENTS.MaxFileSize=30MB** – The maximum size of the log file.

When the file reaches this size, a number is added to its name (for example, **events.log.1**), and a new file is created with the original file name (**events.log**).

- **log4j.appender.EVENTS.MaxBackupIndex=3** – The number of backup log files that are created.
- **log4j.appender.EVENTS.layout=org.apache.log4j.PatternLayout** – Determines whether the log output is formatted by a certain layout.
- **log4j.appender.EVENTS.layout.ConversionPattern=<%d> [%-5p] [%t] - %m%n** – The layout format of the log output.
- **log4j.appender.EVENTS.Threshold=DEBUG** – The output level (can be DEBUG, INFO, WARNING, ERROR, or FATAL).

This definition level filters out log events with a narrower scope than the value of the threshold option. For example, if this definition is set to DEBUG, all entries are logged; if the definition is set to FLOW, all entries apart from DEBUG are logged; if the definition is set to WARNING, all entries apart from FLOW and DEBUG are logged, and so forth.

Reducing Log Load

To reduce log load, you can replace the configurable log files with a predefined file that minimizes the log output. This file defines two log outputs: a console at an ERROR threshold level, and a general file at an INFO level.

To reduce log load:

- 1** Change the name of the configuration file log file, whose operation you want to reduce.
- 2** Copy the **mam4j-non-debug.properties** file, located in:
`\Mercury\<Mercury Application Mapping root directory>\root\lib\server`
to the configuration file location.
- 3** Change the name of the copied file to the original configuration file name.
- 4** Restart the Server.

Defining Server Login Data

The Address list in the Login page displays a list of Servers to which users connect when working with Mercury Application Mapping. You define Server login data in the **server.properties** configuration file, located in:
`\Mercury\<Mercury Application Mapping root directory>\root\lib\gui`

The Server definitions you enter during the installation process appear in this file. You can change these initial definitions and add other Servers to the list. For each Server that appears in the Address list, the following definitions should be set:

- **Server name** – the Server name that appears on the list
- **address** – the Server address

For example:

```
<login>  
  <proxy name="server3" address="http://10.0.64.100:7001/MAM-V3.0"/>  
  <server name="appClient" address="http://10.0.64.199:7001/MAM-V3.0"/>  
</login>
```

or:

```
<login>  
  <proxy name="server3" address="http://10.0.64.100:7001/MAM-V3.0"/>  
  <server name="appClient" address="http://10.0.64.199:7001/MAM-V3.0"/>  
</login>
```

7

Topology Database Management

This chapter describes how to build the Mercury Application Mapping's topology database, how to back it up, how to move it to a different location, how to import and export files, how to save configurations, and how to add objects and links..

This chapter describes:	On page:
Building the Topology Database	143
Exporting the Topology Database	144
Exporting the Topology Database Automatically	145
Importing a Database File	146
Saving and Loading User-Defined Configurations	147
Adding Objects to the Topology Database	148

Building the Topology Database

The **OnlineDBCreator.cmd** script is a command file which builds the topology database from scratch, including tables, definitions, and default configurations. If a database already exists when **OnlineDBcreator.cmd** is activated, **OnlineDBcreator.cmd** deletes the database and builds a new one.

OnlineDBcreator.cmd is automatically activated when you choose to create a new database during the first installation of the Mercury Application Mapping Server. For details on installing the database, see "Creating a Database or Using an Existing Database" on page 74, in *Mercury Application Mapping Installation Guide*.

To build a new database using OnlineDBCreator.cmd:

Double-click the **OnlineDBCreator.cmd** file located in:

\Mercury\<Mercury Application Mapping root directory>\scripts

When creating a new database, **OnlineDBcreator.cmd** scans the files in the subfolders located in:

Mercury\<Mercury Application Mapping root directory>\root\lib\server\db, as follows:

- **acl** – contains default Access Control List (ACL) definitions.
- **classes** – contains definitions for each class, including the name, its inheritances, key attributes, and so forth.

These definitions construct the database scheme and create the class model used by the system.
- **create_objects** – contains definitions of the default objects created by **OnlineDBcreator.cmd**.
- **create_vectors** – contains definitions of the default vectors (a collection of objects that are gathered together in one group) created by **OnlineDBcreator.cmd**.
- **relationships** – describes the available connections (links) between classes defined in the classes folder.

These links are the potential interdependencies that exist between classes. The links appear in the Add Links dialog box during the creation of TQLs.
- **types** – contains all type definitions in the class model.

Exporting the Topology Database

You can export the topology database and move it (or a copy of it) to a different location, using a “hot export” or a “cold export” method:

- **Hot export** – the Server must be up and the export method exports current active events
- **Cold export** – the Server can be up or down as the export method does not depend on the Server and does not export the current active events

To export the database:

- 1 Decide on which export method you are using, and open either **cold_export.cmd** or **hot_export.cmd** in a text editor. The files are located in: **\Mercury\<Mercury Application Mapping root directory>\scripts\backup**
- 2 Locate the **set file_name=.Dmp** definition and enter the name and the location where the file should be saved. Enter the full path, for example:
set file_name=D:\Database_Backup\backup1.dmp
- 3 Save the file.
- 4 Double click the file to run the command. In the DOS window that is displayed, answer **Y** to the question and press **Enter**.

Exporting the Topology Database Automatically

Mercury Application Mapping includes a scheduler that exports an existing database to an external file on a regular basis. Scheduler definitions are set in the **appilogConfig.properties** file, located in:

\Mercury\<Mercury Application Mapping root directory>\root\lib\server

To export the Topology Database automatically:

- 1 Open the **appilogConfig.properties** file in a text editor.
- 2 Locate the Export database properties section in the file:

```
# ----->
#      Export database properties
appilog.general.ExportDatabaseIntervalUnits=day
appilog.general.ExportDatabaseInterval=7
appilog.general.ExportDatabaseNumberOfBackups=10
appilog.general.ExportDatabaseLocation=C:/Mercury/MAM-V3.0/root/dbscripts
appilog.general.ExportDatabaseConnect=appilog/appilog@SKAZAL
appilog.general.ExportDatabaseFirstTime=23:00:00
```

- 3 Edit the definitions as follows:
 - **appilog.general.ExportDatabaseIntervalUnits=day** – the time unit for the backup performance (day, week or month).
 - **appilog.general.ExportDatabaseInterval=7** – the number of time units that passes between each backup. The default is one week.

- **appilog.general.ExportDatabaseNumberOfBackups=10** – the number of backups that are to be performed.
 - **appilog.general.ExportDatabaseLocation=C:/Mercury/<Mercury Application Mapping root directory>/root/dbscripts** – the location of the backup file.
 - **appilog.general.ExportDatabaseConnect=appilog/appilog@SKAZAL** – the connection string to the Oracle database, consisting of a user name, password, and database name (SID).
- 4** Save the file.

Importing a Database File

You can import a database file that you previously exported or you can import a new database file. The Import option cleans the existing database and replaces it with a new database file. If you import a file that you previously exported, you must use the same Oracle database versions for both actions.

To import a database file:

- 1** Open the **import.cmd** file in a text editor. The file is located in:
\Mercury\<Mercury Application Mapping root directory>\scripts\backup
- 2** Locate the **set file_name** definition and enter the name and the location of the database file. Enter the full path, for example:

```
set file_name=D:\Database_Backup\backup1.dmp
```
- 3** Locate the **set path** definition, and verify that the path to the Oracle database is correct, for example:

```
set path=%path%;D:\Oracle\Ora81\BIN
```
- 4** Save the file.
- 5** Double click the file to run the command. In the DOS window that is displayed, answer **Y** to the question and press **Enter**.

Saving and Loading User-Defined Configurations

When you export your database file, the entire database contents are saved, including the configurations defined by the user after the initial creation of the database. These user-defined configurations include TQL queries, managed views, correlation rules, system reports, action rules, time rules, groups, and users.

You can save the user-defined configurations only and recreate the other parts of the database from scratch (for example, when upgrading an application). You can use command files to save user-defined configurations in separate files and load them into the database at a later stage.

This section contains the following topics:

- ▶ “Saving User-Defined Configurations” on page 147
- ▶ “Changing the Location of the User-Defined Configurations” on page 147

Saving User-Defined Configurations

You can save user-defined configurations.

To save user-defined configurations:

Double-click **SaveConfiguration.cmd** to run the command. The file is located in:

`\Mercury\<<Mercury Application Mapping root directory>\scripts\utils`

The configuration files are saved to the location specified in this file.

Changing the Location of the User-Defined Configurations

You can change the location of the user-defined configurations.

To change the location of the user-defined configurations:

- 1** Open **SaveConfiguration.cmd** in a text editor.
- 2** Locate the line
`call .\batch.cmd appilog.server.utils.db.SaveConfiguration D:\\Mercury\\.`
- 3** Change the location and save the file.
- 4** Double click the file to run the command.

Adding Objects to the Topology Database

Objects can either be added to the topology database:

- ▶ through the discovery process – for more information, see “Inserting Objects Through the Discovery Process” on page 148
- ▶ manually – for more information, see “Manual Insertion of Objects” on page 148

You can then add a link between existing objects that are displayed in the Map View – for more information, see “Adding Links Between Existing Objects” on page 151

Inserting Objects Through the Discovery Process

The Mercury Application Mapping discovery process is the mechanism that enables you to collect data about your system by discovering the IT infrastructure resources and their interdependencies. It discovers objects such as, applications, databases, network devices, different types of servers, and so on. Each discovered IT object is then delivered and stored in the topology database where it is represented as a managed object. For more information on the discovery process, see *Mercury Application Mapping Discovery Process Tutorial*.

Manual Insertion of Objects

You can manually insert objects into the topology database that will not be automatically discovered. For example, an object that is located on a remote FTP site might not be included in the system’s discovery scope, and, therefore, in the database.

A new object can be manually added in two different contexts. You can:

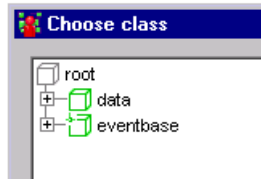
- ▶ Add an object to an existing object that is displayed in Map View, which serves as the added object’s container.
- ▶ Add an object to a view without creating a dependency on an existing object.

Adding an Object to an Existing Object

This section describes how to add an object to an existing object.

To add an object to an existing object:

- 1 In the Map View, in the Edit menu, select **Insert Object** or right-click an object and select **Insert Object** to open the Insert Object dialog box.
- 2 Click the button at the right end of the **Object Class** box to open the Class Model tree.



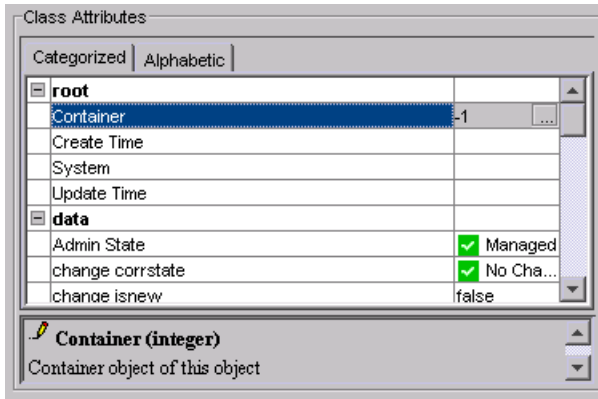
- 3 In the Class Model tree, select the class to which you want to add the new object.

For example, select **IP** in the Class Model tree to add a new IP address object to the database.

In the Insert Object dialog box, the **Class Attributes** section lists the classes contained in this view and their attribute values.

- The classes in the **Alphabetic** tab are listed in alphabetical order.
- The classes in the **Categorized** tab are grouped according to the categories defined in the Class Browser.

- 4 If you are using the **Categorized** tab, click the **Expand** button. to view all the classes contained in this view.



- 5 Edit the attributes as required. You can only edit the attribute if a pencil icon appears at the bottom next to the attribute name. To edit the value of an attribute, click the value cell and then click the square button on the right.
- 6 Click **OK** to save your changes and close the Insert Object window.

Tip: If you relate the new object to an existing object, and the new object does not appear in the view, verify that the view's TQL definitions include the new object. Use the Find option to search for the object, or select the container object and use the Get Neighbors option to display the connected objects. For details, see “Displaying Interdependent Objects”, in *Mercury Application Mapping User's Guide*.

Adding an Object or link to a View without Creating a Dependency on an Existing Object

This section describes how to add an object or link to a view without creating a dependency on existing object.

To add an object or link to a view without creating a dependency on an existing object:

- 1** In the Map View, select **Edit > Insert Object** to open the Insert Object dialog box.
- 2** Follow steps 2 to 6 in “Adding an Object to an Existing Object” on page 149.

Adding Links Between Existing Objects

You can add a link between existing objects that are displayed in the Map View. You can only insert a link between two objects from the same view.

To insert a link between existing objects:

- 1** With the Map View displayed, select the two objects you want to link. For details on how to select multiple links, see “Selecting Multiple Objects” in the *Mercury Application Mapping User's Guide*.
- 2** Right-click the selected objects and select **Insert Link** to open the Insert Link dialog box.
- 3** Follow steps 2 to 6 in “Adding an Object to an Existing Object” on page 149.

Tip: If the link does not appear in the view, verify that the view's TQL definitions include the new link.

8

Customized Package Creation

This chapter explains how to create customized packages to suit your IT management needs.

This chapter describes:	On page:
What is a Package?	154
Package Structure	154
Package Location	157
Package Deployment	157
Dependencies Among Packages	157
Creating Customized Packages	160
Creating a Package	161
Verifying the Validity of a Customized Package	162
Uninstalling and Updating a Customized Package	164

About Creating Customized Packages

Mercury Application Mapping modular packaging enables you to install and work with resources and tools that suit your IT management needs. This allows you to focus only on information and tasks that are meaningful for your organization's IT environment. You can expand, update, or remove existing packages as well as create your own packages based on the resources and tools you customized and developed.

The following packages—**Basic.zip**, **Network.zip** and **Network.zip**—make up the basic installation. **Basic.zip** and **Network.zip** are installed automatically after installing and launching the Mercury Application Mapping server. These packages contain the default definitions, basic resources and tools that are needed for working with the Mercury Application Mapping environment. **Network.zip** is automatically installed with the package installation. All other packages, which are divided according to different categories and technologies, are optional. For more details, see “Installing Mercury Application Mapping Predefined Packages” in the *Mercury Application Mapping Installation Guide*.

What is a Package?

A package is a zip file containing resources that are structured in organized, predefined subdirectories. The subdirectory structure is defined by a file called **packaging.xml**, which is located in: `\Mercury\<Mercury Application Mapping root directory>\root\lib\server`.

Package Structure

Packages can contain any of the following resources:

- **classes** – Contains class descriptions, such as **host**, **file**, **switch**, and so forth.

For details, see , “Introduction to the Class Browser,” in *Mercury Application Mapping User's Guide*.

- ▶ **relationships** – Contains all possible valid connections (links) between the classes that are defined in the classes folder.

For details, see , “Introduction to the Class Browser,” in *Mercury Application Mapping User’s Guide*.
- ▶ **enums** (enumeration definitions) – Contains the definitions that relate to attributes of enumeration type, such as severity levels, admin states and so forth.

For details, see “Building the Topology Database” on page 143.
- ▶ **patterns** (discovery patterns) – Contains the required discovery patterns for the package’s classes and links.

For details on discovery patterns, see “Activating a Discovery Pattern” on page 76.
- ▶ **pql** (TQL) – Contains TQLs that are part of the package,

For details on TQLs, see Part II, “Topology Query Language,” in *Mercury Application Mapping User’s Guide*.
- ▶ **path** (Path Manager) – Contains Path Manager definitions that are part of the package.
- ▶ **correlation** (correlation rules) – Contains correlation rules that are part of the package.

For details on correlation rules, see Part VI, “Object Relationships,” in *Mercury Application Mapping User’s Guide*.
- ▶ **serverLogic** (Server Logic Rules) – Contains server logic rules that are part of the package.

For details on server logic rules, see , “Defining Logical Objects and Rules,” in *Mercury Application Mapping User’s Guide*.
- ▶ **scheduler** – Contains scheduled tasks that are executed in the context of the package.

For details, see Chapter 9, “Task Scheduling.”
- ▶ **reports** – Contains report descriptions relating to the package.

For details, see Part VIII, “System Reports,” in *Mercury Application Mapping User’s Guide*.

- ▶ **view** (View Manager) – Contains View Manager definitions that are part of the package.

For details on the View Manager, see Part III, “Views, Organization Rules, and Notifications,” in *Mercury Application Mapping User’s Guide*.

- ▶ **eventRules** – contains event rule descriptions relating to the package

For details on event rules, see “Defining an Event Rule” on page 24.

- ▶ **timeRules** – contains time rule definitions relating to the package

For details on time rules, see “Configuring a Time Rule” on page 34.

- ▶ **job** – contains job definitions that are part of the package

- ▶ **world** – contains object state holder descriptions in XML format that enable you to create object instances

Resource descriptions use XML format. The resources can be expanded using Mercury Application Mapping.

The following is an example of the description of a host instance using XML format.



Note: After changing resources such as icon properties through the packaging mechanism, you must restart Mercury Application Mapping.

Package Location

When you install Mercury Application Mapping's packages, all packages are placed in the following directory:

`\<Mercury Application Mapping root directory>\root\lib\packages`

All packages must be located in this path. After installing the package file, it is recommended that you set the file to open as read-only in order to prevent any unintentional changes from being made to the file.

Note: The `\Mercury\<Mercury Application Mapping root directory>\root\lib\packages\packagesbu` folder is for Mercury use only. Do not delete this folder.

Package Deployment

Once you have installed the package, the package resources are imported into the corresponding Mercury Application Mapping managers\tools. For example, all the resources you defined in the classes directory will be placed in the Class Browser with the same hierarchical structure.

To verify deployment validity, see the `\Mercury\<Mercury Application Mapping root directory>\root\logs\packaging.log` file.

Dependencies Among Packages

Certain packages are dependent for their functioning on the installation of other packages. These dependencies are specified in the `descriptor.xml` file included in each package. When you install the default packages, Mercury Application Mapping automatically specifies the interdependencies in the `descriptor.xml` file.

For example, the **SQL_Server** package is based on resources that are contained in the **Database_Basic** package. To view this dependency, open **SQL_Server.zip** and open **descriptor.xml** in a text editor:

```
<descriptor>
<dependency>Database_Basic.zip</dependency>
</descriptor>
```

The following table contains a list of the default packages in the package directory and their dependencies.

Package	Dependent on...
Basic	None
Citrix	Host_Resources_Basic
Database_Basic	Host_Resources_Basic, Network
DB2	Database_Basic
Exchange_Resources_By WMI	Database_Basic
F5	Network
FTP	Network
Host_Resources_Basic	Network
Database_Basic	Exchange Resources by WMI
Host_Resources_By_Big_Brother	Host_Resources_Basic
Host_Resources_By_NTCMD	Host_Resources_Basic
Host_Resources_By_SNMP	Host_Resource_Basic
Host_Resource_By_Telnet	Host_Resource_Basic
Host_Resource_By_TTY	Host_Resources_Basic
Host_Resource_By_WMI	Host_Resource_Basic
IBM_HTTP_Server	WebServer, J2EE-JSR77
IIS_Resources_By_WMI	Network, WebServer
J2EE-JSR77	Database_Basic

Package	Dependent on...
Layer2	Network
LDAP	Network
Network	Basic
Oracle	Database_Basic
P2P	Network, Host_Resources_Basic
PathManager	Network
Rules	Basic
SAP	Network, Database_Basic
Siebel	Host_Resources_By_Wmi, Database_Basic, WebServer
SQL_Server	Database_Basic
Sybase	Database_Basic
TCP_discovery	Network
Weblogic61Jar	J2EE-JSR77
Weblogic70Jar	J2EE-JSR77
WebServer	Network
Websphere_MQ	Network, Host_Resources_By_Telnet

Creating Customized Packages

You can create a customized package to meet the needs of your organization's home-grown applications.

When creating a package, the same top-level structure and names must match the definitions described in the **packaging.xml** file to maintain consistency with the deployment mechanism. You can change the lower levels as required, that is, you can add resources and folders.

Important: It is highly recommended to verify the validity of a customized package before actually deploying it (for details, see “Verifying the Validity of a Customized Package” on page 162).

XML File Naming Conventions

This section describes the naming conventions for XML file name for the resources whose manager is a folder according to **packaging.xml**. The resources are:

- ▶ TQL
- ▶ Path Manager
- ▶ Correlation Rules
- ▶ Server Logic Rules
- ▶ View Manager
- ▶ Reports

Following are the naming conventions:

- ▶ The name of the resources described in the XML file must be identical to the name of the XML file. For example, a correlation rule called **MyCorrelation** should be described in a file called **MyCorrelation.xml**.
- ▶ The XML file name is case sensitive. For example, if the name of the resource is called **timeRules**, then the name of the XML file must be **timeRules.xml** and not **timerules.xml** or **TimeRules.xml**.

Creating a Package

You can create your own customized package.

To create a package:

- 1** Create the necessary folder or folders as defined in `\Mercury\<Mercury Application Mapping root directory>\root\lib\server\packaging.xml`.
- 2** Verify the validity of the package (see “Verifying the Validity of a Customized Package” on page 162).
- 3** Place the relevant resources in the corresponding directories. For example, If you create a correlation rule, place it in the Correlation directory.
- For a list of the resources you can place in the package, see “Updating Customized Packages” on page 165.

Important: All dependencies must be specified in the `descriptor.xml` file of the package.

- 4** Zip the directories you have created, including the `descriptor.xml` file.
- 5** Place the zip file you have created in `\Mercury\<Mercury Application Mapping root directory>\root\lib\packages`. The scheduler automatically deploys the packages in their correct managers and places the corresponding resources into their matching directories.

Verifying the Validity of a Customized Package

You can simulate the package deployment process to verify how a customized package affects the system.

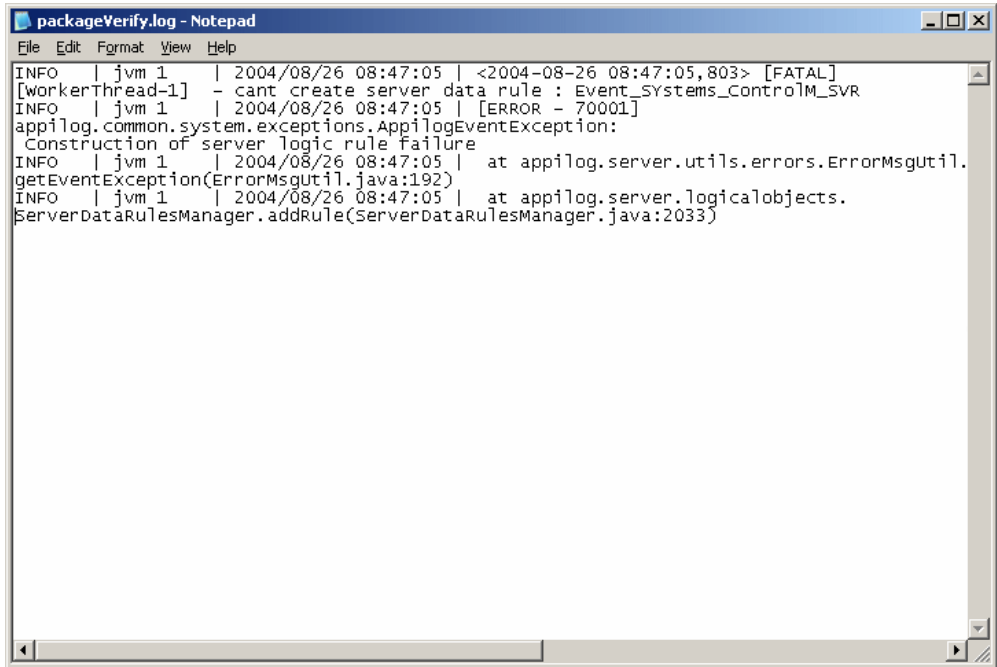
You use the **packageVerify.cmd** file to validate a customized package. This file checks the validity of the package structure, syntax, and dependencies. For example, if the package depends on a class that does not exist in the system, validation fails.

The **packageVerify.cmd** file performs the following:

- ▶ Verifies that the package structure of the customized package reflects the package structure as defined in the **packaging.xml** file.
- ▶ Verifies the syntax in the corresponding XML file of each resource.
- ▶ Simulates the effect that removing a resource has on the system once a package is deployed. For example, if a class is removed, all other classes or resources that depend on this class are also removed.
- ▶ Adds the resources in the deployed packages to the Mercury Application Mapping database.

Validation messages are written to the **packageVerify.log** file. If validation is successful, the log displays a message indicating that the package has been deployed successfully.

If validation fails, the log displays a message indicating the reason for the failure, as seen in the following example:



```

packageVerify.log - Notepad
File Edit Format View Help
INFO | jvm 1 | 2004/08/26 08:47:05 | <2004-08-26 08:47:05,803> [FATAL]
[workerThread-1] - cant create server data rule : Event_Systems_ControlM_SVR
INFO | jvm 1 | 2004/08/26 08:47:05 | [ERROR - 70001]
appilog.common.system.exceptions.AppilogEventException:
Construction of server logic rule failure
INFO | jvm 1 | 2004/08/26 08:47:05 | at appilog.server.utils.errors.ErrorMessageUtil.
getEventException(ErrorMessageUtil.java:192)
INFO | jvm 1 | 2004/08/26 08:47:05 | at appilog.server.logicalobjects.
ServerDataRulesManager.addRule(ServerDataRulesManager.java:2033)

```

Once validation is successful, you can install your package.

To simulate package deployment:

- 1** Copy the packages you want to validate to the following location:
\Mercury\<Mercury Application Mapping root directory>\root\lib\server\verify.
- 2** Run the **packageVerify.cmd** file located in:
\Mercury\<Mercury Application Mapping root directory>\scripts\manualScripts
- 3** View the validation messages that are written to the **packageVerify.log** file located in:
\Mercury\<Mercury Application Mapping root directory>\root\logs\packageVerify.log

Uninstalling and Updating a Customized Package

This section describes how to uninstall and update a package and specifies what considerations to take into account before performing these actions.

Before updating or deleting a resource, take the following considerations into account:

- ▶ If you delete a package or any of the resources inside the package, those resources are deleted from the database.
- ▶ Before deleting a class that has inheriting classes, delete its inheriting objects and then delete the class.
- ▶ Before redeploying a package that has not undergone any changes, change the timestamp in the package file to a later date.
- ▶ While creating a package, take into account that some classes might be connected by inheritance. To maintain the inheritance between them, place the classes in the directory from which they inherited their attributes.
- ▶ Before making a change to a package, (adding, removing, or updating) close Mercury Application Mapping and log in again.

This section contains the following topics:

- ▶ “Uninstalling Customized Packages” on page 164
- ▶ “Updating Customized Packages” on page 165

Uninstalling Customized Packages

You can uninstall customized packages.

To uninstall customized packages:

- ▶ Delete the package zip file from the `\Mercury\<Mercury Application Mapping root directory>\root\lib\packages` directory.

Caution: Deleting a package removes all the resources from the database.

Updating Customized Packages

You can update customized packages.

To update customized packages:

- 1 Open the package's zip file.
- 2 Make the required changes.
- 3 Repeat 4 and 5 in "Creating a Package" on page 161.

Caution: Bear in mind that activating **OnlineDBCreator.cmd** results in the deployment of the default packages and thus overrides any changes that have been made – for details on **OnlineDBCreator.cmd**, see "Building the Topology Database" on page 143.

9

Task Scheduling

This chapter describes how you can use the Mercury Application Mapping Scheduler to execute tasks to run on a periodic basis.

This chapter describes:	On page:
Predefined Tasks	168
Creating a Task	169
Activating Tasks	170
Scheduling a Task	170
Scheduled Actions	175

About Task Scheduling

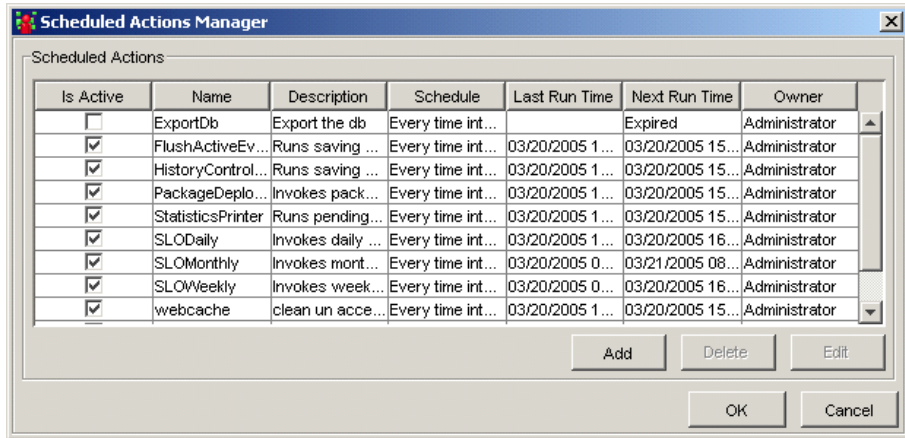
Mercury Application Mapping enables you to define tasks that will be activated on a periodic basis. For example, you can define a schedule for an automatic backup of the reports that are generated, and determine how many backups will be saved.

The procedure for scheduling tasks is as follows:

- 1 Create a task (for details, see “Creating a Task” on page 169).
- 2 Schedule a task (for details, see “Scheduling a Task” on page 170).

Predefined Tasks

The Scheduled Actions Manager contains a set of predefined, commonly used tasks:



Note: You cannot modify the predefined tasks.

The predefined tasks Mercury Application Mapping provides are:

- ▶ **ExportDb** – exports the database.
- ▶ **FlushActiveEvent** – saves the events that are currently active in the system's memory to the database.
- ▶ **HistoryController** – saves changes of attribute values to the system's database. For internal use only.
- ▶ **PackageDeployer** – deploys and updates packages.
- ▶ **StatisticsPrinter** – writes statistics to the log file.
- ▶ **SLODaily** – calculates the daily service level availability.
- ▶ **SLOMonthly** – calculates the monthly service level availability.
- ▶ **SLOWeekly** – calculates the weekly service level availability.
- ▶ **webcache** – an internal cache mechanism. For internal use only.

- **activateRemoteDuplicateHosts** – removes duplicate hosts.
- **ClearHistory** – clears the history database. For internal use only.

Creating a Task

The following procedure explains how to create a task.

To create a task:

- 1** Select **Administration > Scheduler** to open the Scheduled Actions Manager.
- 2** Click **Add** to open the Scheduled Action Wizard.
- 3** Enter a unique name for the task in the **Name** box.
- 4** (Optional) Enter a description of the task in the **Description** box.
- 5** Click **Add** to add actions to the task. The Actions dialog box displays a list of actions that can be executed by the Scheduled Actions Manager.
- 6** Choose the required action from the list. For a description of the actions contained in the **Actions** box, see “Scheduled Actions” on page 175.
- 7** Click **Next** to display a list of rules for the selected action. At this stage the procedure depends on which action you choose.

For example, if you select **Send Event on TQL Results**, you must select a TQL name, and define an event ID, a message, and the severity level.
- 8** Click **Finish**. The action appears in the Scheduled Action Wizard.
- 9** To define another action, click **Add** and repeat steps 6 to 8.
- 10** Click **Next** to define when you want the action or actions to be performed (for details, see step 4 in the next section).

To edit a task:

- 1** Select **Administration > Scheduler** to open the Scheduled Actions Manager.
- 2** Select the task you want to edit and click **Edit**.
- 3** Click the **Task** tab to edit the task information (for details, see steps 3 to 6 in the previous procedure).

- 4 Click the **Schedule** tab to edit the scheduling information (for details, see “Scheduling a Task” on page 170).
- 5 Click **OK** to save the changes.

Activating Tasks

The following procedure describes how to activate a task.

To activate a task:

- 1 Select **Administration > Scheduler** to display the Scheduled Actions Manager dialog box.
- 2 Locate the action you want to activate and select the **Is Active** check box.
- 3 Click **OK** to save the change.

Scheduling a Task

This section explains how to set the schedule for activating a task. It has the following topics:

- “Scheduling Tasks” on page 171
- “Running a Task on a Daily Basis” on page 172
- “Running a Task on a Weekly Basis” on page 173
- “Running a Task on a Monthly Basis” on page 173
- “Running a Task Once” on page 174
- “Running a Task at Set Time Intervals” on page 174
- “Selecting the Range of Recurrence” on page 175

Scheduling Tasks

You can schedule a task.

To schedule a task:

If you are scheduling a task directly after creating it, skip to step 4.

- 1** Select **Administration** > **Scheduler** to open the Scheduled Actions Manager.
- 2** Select the action which you want to schedule and click **Edit** to open the Edit Scheduled Actions dialog box.
- 3** Click the **Schedule** tab.
- 4** Select one of the following options:
 - ▶ **Daily** – activates the action on a daily basis (for details, see “Running a Task on a Daily Basis” on page 172).
 - ▶ **Weekly** – activates the action on a weekly basis (for details, see “Running a Task on a Weekly Basis” on page 173).
 - ▶ **Monthly** – activates the action on a monthly basis (for details, see “Running a Task on a Monthly Basis” on page 173).
 - ▶ **One Time Only** – activates the action only once (for details, see “Running a Task Once” on page 174).
 - ▶ **Every Time Interval** – activates the action at a predefined time interval (for details, see “Running a Task at Set Time Intervals” on page 174).
- 5** Click **OK** to save the task. You are returned to the Scheduled Actions Manager.

Running a Task on a Daily Basis

You can run a task on a daily basis.

To perform a task on a daily basis:

- 1** If you select **Daily** in the Scheduled Actions Wizard (see step 4 in “Scheduling a Task” on page 170), you must select the time and day on which you want to activate the action.
- 2** To choose a time (hours, minutes, and seconds) at which you want to activate the action, select **Time** and click the down arrow to the right of the **Time** box. Use the diagonal buttons to set the required time. Click the down arrow again to close the **Time** box.

To choose a time (hours only) at which you want to activate the action, select **Hours** and enter the time in the Hours box, as follows:

- ▶ **1 to 12** – represent the hours between 1:00 AM and 12:00 noon
- ▶ **13 to 24** – represent the hours between 1:00 PM and 24:00 midnight

To specify that you want a task to run consecutively, use a hyphen between two numbers. For example, to specify 8:00 PM to 11:00 PM, enter **20:00-23:00**.

To specify times that do not run consecutively, use a comma. For example, to specify that you want the action to run at 8:00 PM, 10:00 PM, and from 2:00 AM to 5:00 AM, enter **20:00, 22:00, 2:00-5:00**.

- 3** Select the range of recurrence (for details, see “Selecting the Range of Recurrence” on page 175).
- 4** Click **Finish** to save the settings.

Running a Task on a Weekly Basis

You can run a task on a weekly basis.

To perform a task on a weekly basis:

- 1** If you select **Weekly** in the Scheduled Actions Wizard (see step 4 in “Scheduling a Task” on page 170), you must select the time and days on which you want to activate the action.
- 2** In the **Time** box, enter the time (hours only) at which you want to activate the action, as follows:
 - ▶ **1 to 12** – represent the hours between 1:00 AM and 12:00 noon
 - ▶ **13 to 24** – represent the hours between 1:00 PM and 24:00 midnight
- 3** Select the day or days on which you want the action to run.
- 4** Select the range of recurrence (for details, see “Selecting the Range of Recurrence” on page 175).
- 5** Click **Finish** to save the settings.

Running a Task on a Monthly Basis

You can run a task on a monthly basis.

To perform a task on a monthly basis:

- 1** If you select **Monthly** in the Scheduled Actions Wizard (see step 4 in “Scheduling a Task” on page 170), you must select the months, days, and time on which you want to activate the action.
- 2** In the **Time** box, enter the time (hours only) at which you want to activate the action, as follows:
 - ▶ **1 to 12** – represent the hours between 1:00 AM and 12:00 noon
 - ▶ **13 to 24** – represent the hours between 1:00 PM and 24:00 midnight
- 3** In the **On Days** box, enter the days on which you want to activate the action. Use the numbers **1 to 31** to represent the days of the month.

To specify that you want an action to run on consecutive days, use a hyphen between two numbers. For example, to specify from the 2nd to the 7th, enter **2-7**.

To specify that the action should run on nonconsecutive days, use a comma. For example, to specify that the action should run on the 8th, the 15th, and the 20th, enter **8, 15, 20**.

- 4 Select the range of recurrence (for details, see “Selecting the Range of Recurrence” on page 175).
- 5 Click **Finish** to save the settings.

Running a Task Once

You can define a task that is scheduled to run once.

To perform a task once:

- 1 If you select **One Time Only** in the Scheduled Actions Wizard (see step 4 in “Scheduling a Task” on page 170), you must select the date and time on which you want to activate the action.
- 2 To choose the date and time when the action should begin running, click the down button in the **Start date and time** box to display a calendar.
- 3 In the **Date** and **Time** tabs, use the diagonal arrow buttons to choose the date and time.
- 4 Click the down button to close the calendar and then click **Finish** to save the settings you have defined.

Running a Task at Set Time Intervals

You can run a task at set time intervals.

To set time intervals for running the action:

- 1 If you select **Every Time Interval** in the Scheduled Actions Wizard (see step 4 in “Scheduling a Task” on page 170), you must select the time intervals at which you want to activate the action.
- 2 In the **Every** field, type a value for the interval between successive runs.
- 3 Choose the required unit of time measurement (seconds, minutes, hours, days, or weeks).

- 4 Select the range of recurrence (for details, see “Selecting the Range of Recurrence” on page 175).
- 5 Click **Finish** to save the settings.

Selecting the Range of Recurrence

You can select the range of recurrence.

To select the range of recurrence:

- 1 To choose the date and time when the action should begin running, click the down button in the **Start** box to display a calendar.
- 2 In the **Date** and **Time** tabs, use the diagonal arrow buttons to choose the date and time.
- 3 Click the down button to close the calendar.
- 4 To choose the date and time when the action should stop, select **End by**, and repeat the procedure for choosing a date and time.
- 5 If you do not want to specify an ending date, select **No End Date**.

Scheduled Actions

The Scheduled Actions Wizard includes a list of actions that Mercury Application Mapping can activate, as follows:

- **Acknowledge all TQL results** – Acknowledge all the active events that apply to all TQL-related objects so that blinking objects stop blinking.
- **Activate a Correlation Rule** – Activate the selected correlation rule.
- **Activate an Event Rule** – Activate the selected event rule.
- **Activate a Logical Object Rule** – Activate a selected logical object rule.

Note: Mercury Application Mapping allows for multiple selection of logical object rules.

- ▶ **Activate a Time Rule** – Activate a selected time rule.
- ▶ **Invoke Discovery Patterns on TQL results** – Add a discovery pattern to all TQL-related objects.
- ▶ **Clear TQL result state** – Deactivate all the active events of all TQL-related objects so that all objects are in a normal state.
- ▶ **Disable a Correlation Rule** – Disable the selected correlation rule.
- ▶ **Disable an Event Rule** – Disable the selected event rule.
- ▶ **Disable a Logical Object Rule** – Disable the selected logical object rule.
- ▶ **Disable a Time Rule** – Disable the selected time rule.
- ▶ **Export the Database** – Export the database to a file.
- ▶ **Remove all Active Events of the TQL results** – Remove the active events of all TQL-related objects so that all objects are in a normal state.
- ▶ **Remove Discovery Pattern from the TQL results** – Delete the selected discovery patterns of all TQL-related objects.
- ▶ **Delete TQL results from the database** – Delete all TQL-related objects.
- ▶ **Generate and save a System Report** – Generate a report and save it to file.
- ▶ **Send events on TQL Results** – Send a raw event for all TQL-related objects.
- ▶ **Set the Admin State of the TQL results** – Set an admin state for all TQL-related objects.
- ▶ **Set an attribute value of TQL results** – Assign an attribute value to all TQL-related objects.
- ▶ **Suppress TQL Result Blinks** – Suppress for all active events that apply to all TQL-related objects so that all blinking objects stop blinking.
- ▶ **Unacknowledge all TQL results** – Unacknowledge all previously acknowledged actions for all the active events that apply to all TQL-related objects so that the objects with active events blink.
- ▶ **Unsuppress TQL Result Blinks** – Clear previously suppressed actions for all active events that apply to all TQL-related objects so that objects with active events blink.

10

Dynamic Object States

This chapter describes how to define the dynamic management state of an object.

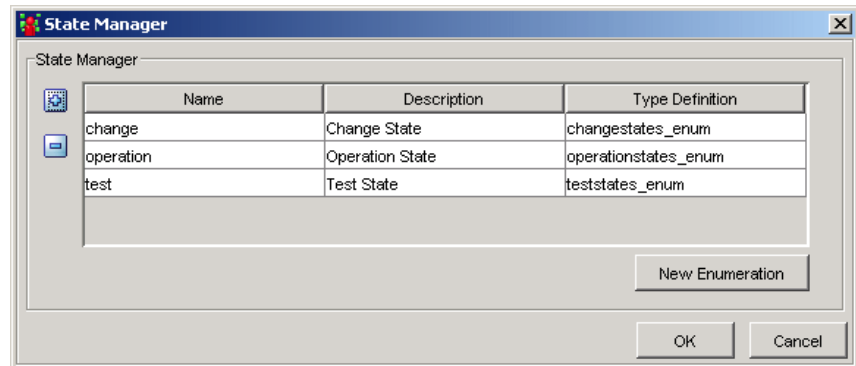
This chapter describes:	On page:
Creating Dynamic Management States	177
Using States	179

Creating Dynamic Management States

You can dynamically define the management state of an object. You can create additional management states for controlling and measuring management aspects, such as performance, security, and so forth.

To create management states:

- 1 Select **Administration > State Manager** to open the State Manager dialog box.



Note: The dialog box appears with default states that cannot be edited in any way.



- 2 Click the expand button to create a new row.
- 3 Enter details of the category as follows:

Field	Description
Name	Enter a unique name for the state.
Description	(Optional) Enter a description of the state.
Type Definition	<p>Click in the Type Definition field to display a list of the Enumerations that can be used to describe the severity/states and values. Select the desired Type Definition.</p> <p>Note: The list contains only Enumeration definitions to which the rules for creating a severity list for states are applied. To create an Enumeration definition, click the New Enumeration button. For details on creating Enumeration definitions, see Chapter 26, “Enumerations and Lists,” in <i>Mercury Application Mapping User’s Guide</i>.</p>

- 4 Click **OK** to save the state. The state you created appears in the State Manager dialog box.

Using States

The states you create become part of the class model. For every object in the system, Mercury Application Mapping manages three state attributes:

- ▶ **<category> corrstate** – manages the correlation state of the object
- ▶ **<category> isnew** – indicates if the object has any new events
- ▶ **<category> state** – manages the object's state

For example, for the state **Performance**, Mercury Application Mapping automatically creates the attributes **Performance corrstate**, **Performance isnew**, and **Performance state**.

These attributes can be used wherever you define attribute conditions for an object.

To view the attributes:

- 1** In the TQL Builder map pane, right-click an object and select **TQL Node Definition** to open the TQL Node Definition dialog box.
- 2** Click **Add** and open the **Attribute Name** list. The list includes the attributes for the state you created.

11

Oracle Configuration, Monitoring, and Tuning Guidelines

This chapter describes the recommended Oracle database configuration, monitoring, and tuning for Mercury Application Mapping.

This chapter describes:	On page:
Oracle Configuration Guidelines	182
Monitoring and Tuning Guidelines	186

About Oracle Configuration, Monitoring, and Tuning Guidelines

Memory, CPU, and I/O are the three most common system resources consumed by a database server. Database and system administrators should proactively maintain and tune the database servers in their sites. Third-party tools can provide a comprehensive solution for monitoring and identifying bottlenecks in the system. This section provides general guidelines for tuning and configuring the Mercury Application Mapping Oracle database server.

Oracle Configuration Guidelines

This section describes the Oracle and RAID configuration settings.

This section contains the following topics:

- ▶ “Oracle Configuration Settings” on page 182
- ▶ “Using RAID Configuration” on page 184
- ▶ “System Global Area (SGA)” on page 185

Oracle Configuration Settings

Mercury Application Mapping database configuration requirements depend on the quantities of data to be generated. For example, for less than 10 MB of managed objects and a database smaller than 1 GB, you should use a small database. For more than 10 MB of managed objects and a database larger than 1 GB, you should use a large database.

The following table contains the recommended Oracle parameters and values for small and large Mercury Application Mapping applications.

Parameter Name	For Small Applications	For Large Applications	Remarks
DB_BLOCK_SIZE	8192	8192	
DB_BLOCK_BUFFERS or DB_CACHE_SIZE	15 KB or 120 MB	31 KB or 150 MB	It is recommended that you use the DB_CACHE_SIZE parameter.
DB_CACHE_ADVICE	ON	ON	
SHARED_POOL_SIZE	55 MB	80 MB	
LOG_BUFFER	512 KB	1 MB	
DB_FILE_MULTIBLOCK_READ_COUNT	16		
PROCESSES	200	200	
SESSIONS	225	225	

Parameter Name	For Small Applications	For Large Applications	Remarks
<code>SORT_AREA_SIZE</code>	1.5	1.5	This parameter is reserved for backward compatibility and shared server mode. For this parameter to work, the <code>WORKAREA_SIZE_POLICY</code> parameter must be set to <code>MANUAL</code> . In version 9i onwards, it is recommended that you use the <code>PGA_AGGREGATE_TARGET</code> parameter instead of this parameter.
<code>SORT_AREA_RETAINED_SIZE</code>	Equal to <code>SORT_AREA_SIZE</code> value	Equal to <code>SORT_AREA_SIZE</code> value	See remarks for the <code>SORT_AREA_SIZE</code> parameter above.
<code>HASH_AREA_SIZE</code> 6	4.5 MB	4.5 MB	Is equal to three times the <code>SORT_AREA_SIZE</code> value. See remarks for the <code>SORT_AREA_SIZE</code> parameter.
<code>WORKAREA_SIZE_POLICY</code>	Auto	Auto	
<code>PGA_AGGREGATE_TARGET</code>	200 MB	250 MB	
<code>STATISTICS_LEVEL</code>	TYPICAL	TYPICAL	
<code>UNDO_MANAGEMENT</code>	AUTO	AUTO	
<code>UNDO_RETENTION</code>	2000	2000	
<code>HASH_JOIN_ENABLED</code>	TRUE	TRUE	

Using RAID Configuration

The use of RAID is transparent to Oracle. All the features specific to RAID configurations are handled by the operating system and not by Oracle. The use of RAID devices differs according to the Oracle file type. Data files and archive logs can be placed on RAID devices, since they are accessed randomly.

Redo logs should not be put on RAID devices since they are accessed sequentially and performance is enhanced by having the disk drive head near the last write location. However, mirroring of redo log files is strongly recommended by Oracle. RAID is much easier to use than the Oracle techniques for data placement and striping.

Note the following RAID configuration recommendations:

- ▶ RAID usually impacts write operations more than read operations. This is especially true where parity needs to be calculated (RAID 3, RAID 5, and so forth).
- ▶ You can place online or archived redo log files on RAID 1 devices. Do not use RAID 5. In addition, place TEMP tablespace data files on RAID 1 devices, instead of RAID 5, because the streamed write performance of distributed parity (RAID 5) is not as efficient as that of simple mirroring (RAID 1).
- ▶ Swap space can be used on RAID devices without affecting Oracle.

The following table describes the RAID devices and types to be used with each Oracle file type:

RAID	Type of Raid	Control File	Database File	Redo Log File/Temporary	Archive File
0	Striping	Avoid	OK	Avoid	Avoid
1	Shadowing	OK	OK	Recommended	Recommended
0+1	Striping and shadowing	OK	Recommended	Recommended (see following notes.)	Recommended (see following notes.)

RAID	Type of Raid	Control File	Database File	Redo Log File/Temporary	Archive File
3	Striping with static parity	OK	Avoid when this data file involves heavy write operations.	Avoid	Avoid
5	Striping with rotating parity	OK	Avoid when this data file involves heavy write operations.	Avoid	Avoid

Notes:

- ▶ RAID 0 does not provide protection against failures. It requires a strong backup strategy.
 - ▶ RAID 0+1 is recommended for database files because it avoids hot spots and provides the best possible performance during a disk failure. The disadvantage of RAID 0+1 is its costly configuration.
 - ▶ The most recommended configuration for temporary or redo logs is RAID 0+1 with a memory mirrored cache (at least two controllers, each LUN accessible via controllers and a synchronized controllers' cache).
-

System Global Area (SGA)

Configure your SGA to fit physical memory and avoid using swapping. It is recommended that you set the SGA for less than 70% of system physical memory so as to leave sufficient memory for additional system and client processes.

Monitoring and Tuning Guidelines

This section describes the monitoring and tuning guidelines.

This section contains the following topics:

- ▶ “Monitoring CPU and I/O” on page 186
- ▶ “Oracle Alert File” on page 186
- ▶ “Archive Log Files” on page 187
- ▶ “Tablespace Storage Space” on page 187
- ▶ “Dictionary-Managed Tablespace Coalescing” on page 188
- ▶ “Collecting Statistics” on page 189
- ▶ “Database Load Behavior” on page 189

Monitoring CPU and I/O

It is recommended that you monitor the CPU and file system—the main resources consumed by the database server. CPU usage should not exceed 70% and the I/O wait should not be higher than 10%. You can use **Perfmon** in Windows, or **top** in UNIX, to monitor these resources and if you have Mercury SiteScope, you can use it to monitor CPU and I/O both in Windows and in Unix.

Oracle Alert File

Oracle registers abnormal events in the **alert.log** file, whose location is defined by the **BACKGROUND_DUMP_DEST** parameter. It is recommended that you check this file regularly to identify abnormalities that should be corrected, such as **ORA-*<number>*** errors.

Archive Log Files

When using the **archive** mode, monitor your ARCHIVE_DUMP_DEST location for disk usage. These files should be backed up and deleted regularly to leave sufficient disk space for new archive files.

The archive file is usually the same size as the redo log file. To determine the size of a redo log file, use the operating system command or the following query:

```
SQL> select GROUP#, BYTES
from V$LOG;
```

To determine how many archive files are generated over a period of time (for example, a day) you can use the following query, after the system is stable:

```
SQL> alter session set NLS_DATE_FORMAT = 'DD-MON-YYYY';
SQL> select TO_DATE(TO_CHAR(FIRST_TIME,'DD-MON-YYYY')) as "Day",
COUNT(*) as "Number of files"
from V$LOG_HISTORY
group by TO_CHAR(FIRST_TIME,'DD-MON-YYYY')
order by 1 asc;
```

Tablespace Storage Space

To avoid space errors caused by data growth, monitor your tablespace usage regularly.

If a tablespace runs out of space, you can add one or more data files to it by using the following command:

```
ALTER TABLESPACE <tablespace name> ADD DATAFILE
```

Dictionary-Managed Tablespace Coalescing

Mercury Application Mapping version 2.4 Service Pack 2 creates the necessary application tablespaces as locally-managed tablespaces (LMT). This guideline relates only to dictionary-managed tablespaces (DMT), if such are used in the database.

Free space in Oracle tablespaces is composed of newly created extents or extents that have been used and are now free. If some of the free space in a tablespace is composed of extents that were used and freed, the tablespace can become temporarily fragmented.

To repair fragmentation, two extents that reside next to one another can be coalesced to create one large extent.

To check for fragmentation, run the following query using SQL*Plus (using the system administrator account):

```
SELECT A.TABLESPACE_NAME, COUNT(*) BLOCK_CASES
FROM DBA_FREE_SPACE A, DBA_FREE_SPACE B
WHERE A.TABLESPACE_NAME = B.TABLESPACE_NAME
AND A.FILE_ID = B.FILE_ID
AND A.BLOCK_ID+A.BLOCKS = B.BLOCK_ID
GROUP BY A.TABLESPACE_NAME
/
```

This query returns a list of tablespaces that require coalescing. Although the Oracle **SMON** process automatically performs coalescing, the process might be performed infrequently. It is therefore recommended that you coalesce tablespace extents using the following command:

```
ALTER TABLESPACE <tablespace name> COALESCE;
```


Collecting Statistics

The Mercury Application Mapping Oracle database server works with the Cost-Based Optimizer (CBO). The Optimizer is responsible for making an “execution plan” for the SQL statements: the best plans are created based on current statistics about the affected database objects. Mercury Application Mapping version 2.4 Service Pack 2 installs a job for collecting daily statistics. The job has to be in “Normal” state at all times to ensure frequent statistics collection.

To collect statistics for all Mercury Application Mapping objects:

- 1 Log in to the application schema using SQL*Plus.
- 2 Run the following command:

```
Exec DBMS_STATS.GATHER_SCHEMA_STATS
(ownname => 'Mercury', , cascade => TRUE);
```

Note: Once the application is stable, you should collect statistics once daily.

Database Load Behavior

Run **STATSPACK** regularly to monitor the database behavior. For additional information on running and interpreting the output you receive, see *Oracle Metalink Note 94224.1: STATSPACK FAQ*.

It is also recommended that you monitor the I/O load on the system to identify I/O contention. Once you determine which disk is most heavily loaded, you can use the **STATSPACK** output to determine which particular Oracle data file is the cause of the contention and consider the relocation of data files.

12

Oracle Backup Guidelines

This chapter describes the recommended Oracle backup strategy for Mercury Application Mapping.

This chapter describes:	On page:
Backup Methods	192
Oracle Recovery Manager (RMAN)	195
Customizing and Running the Cold Backup Script	197
Backup Scripts	197

About Oracle Backup Guidelines

This chapter describes how to back up the Oracle database.

Your backup strategy is tested when a failure occurs and data is lost. You can lose or corrupt data in several ways, such as a logical application error, an instance failure that prevents Oracle from starting, or a media failure caused by a disk crash.

In addition to your scheduled backups, it is important to perform a backup when the database structure changes (for example, when a data file is added to the database), or before you upgrade your software or hardware.

When choosing a backup strategy, consider several factors, such as the system workload, the usage schedule, the importance of the data, and the hardware environment of the database.

You perform Oracle backups with scripts executing SQL commands as well as operating system commands that copy files. You can also use Oracle Recovery Manager (RMAN) commands.

It is recommended that you maintain updated records of backups performed on your database so that you can use them for recovery on demand. If you are using RMAN, catalog information is available from the catalog.

Backup Methods

This section describes several backup methods: cold, hot, or export.

This section contains the following topics:

- “Cold Backup” on page 192
- “Hot Backup” on page 193
- “Export” on page 193

Cold Backup

Cold backup, also known as offline backup, is a database level backup. This backup method enables recovery to a point in time in the past at which the database snapshot was taken. Usually, the database should be shut down before backup is begun. The length of downtime is dependent on the database size, the backup media (disk or tape), the backup software, and the hardware in use.

Once the instance is down, its data files, log files, control files, and configuration files should be copied either to disk or to another media. After copying has completed, the instance can be restarted.

For the procedure for backing up a database with a cold backup, see “Customizing and Running the Cold Backup Script” on page 195.

For details on backing up Oracle, see the *Oracle Backup and Recovery Guide*.

Hot Backup

Hot backup, also known as online backup, enables you to run a backup while the instance is running and users are connected to the database. This method enables recovery to any point in time. Note that working in archivelog mode requires additional disk space to contain incremental archive files that can influence database performance.

This backup method works at tablespace backup level and requires the database to operate in archivelog mode, enabling Oracle to track changes over time by generating redo log file copies called archive files.

The generated archive files are written to the archive destination specified by the LOG_ARCHIVE_DEST (or LOG_ARCHIVE_DEST_NN) parameter in the instance parameter files. Other related archiving parameters are LOG_ARCHIVE_FORMAT and LOG_ARCHIVE_START.

Once you begin the backup, the data files, control files, archive files, and configuration files should be copied either to disk or to another media.

During the backup process, Mercury Application Mapping can experience performance degradation due to disk load.

For details on backing up Oracle, see the *Oracle Backup and Recovery Guide*.

Export

The export backup utility is a logical backup method that dumps schema structure and contents into an Oracle structured file. This method can be used to transfer data between two schemas in the same database, or between two separate Oracle databases. To load exported data back into the database, use the import utility.

For details, see the Export/Import section of *Oracle Utilities*.

Oracle Recovery Manager (RMAN)

Oracle Recovery Manager (RMAN) is a component of the Oracle database that provides a tightly integrated method for creating, managing, restoring, and recovering Oracle database backups.

You can choose to work with the RMAN catalog schema. The catalog is managed within the Oracle schema and stores information on the registered database structure and backups performed using RMAN. The catalog can be queried to produce backup reports and copy availability. A single catalog can manage backup information from one or more target databases.

The RMAN catalog is usually placed on a different database instance to the operational database and has a backup strategy of its own. The RMAN catalog needs to be available only during the backup or recovery process.

The RMAN tool can be used in conjunction with third-party backup software for a complete backup and recovery solution.

Some of RMAN advantages are:

- ▶ Minimizes backed up data by compressing backed up files to exclude empty data blocks, thereby saving time and space.
- ▶ Supports incremental backups.
- ▶ Supplies the user with backup status reporting ability.
- ▶ Supports parallel backup and recovery processes when possible.
- ▶ Can be used with a third-party backup media tool.

For details on RMAN, see the *Oracle Recovery Manager User's Guide* and *Oracle Recovery Manager Reference*.

Customizing and Running the Cold Backup Script

Mercury Application Mapping supplies a batch backup script that builds a customized batch file. You run the batch file to back up the database in cold backup mode. For details, see “Cold Backup” on page 192.

This section contains the following topics:

- “Windows Environment” on page 195
- “UNIX Environment” on page 196

Windows Environment

The following procedure explains how to back up an Oracle database in cold mode in a Windows environment.

To customize and run the cold backup script:

- 1** Copy the script in “The coldbackup.cmd Script for Windows Environment” on page 198 to a text editor. Save the file as **coldBackup.bat**.
- 2** Open the **coldbackup.bat** file in a text editor and change the following lines to reflect your Oracle home directory and Oracle password:

```
REM !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
REM These values cannot be derived. Set them to reflect your environment
REM !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
set ORACLE_HOME=C:\Oracle\ora92
set ORACLE_SID=ORCL01
set O_CONNECT="sys/sys as sysdba"
set O_INIT=C:\Oracle\ADMIN\ORCL01\pfile\init.ora
set O_PWFIL=%%ORACLE_HOME%\database\PWDORCL01.ORA
```

- 3** Open a Command Prompt window and run the following command:

```
coldBackup.bat backup_script_location backup_files_location
```

where

- **backup_script_location** – the location of the back up scripts
- **backup_files_location** – the location to which the database files should be backed up

Example:

```
c:\cold> coldBackup.bat c:\coldbackupscript f:\backupfiles
```

The cold_backup.cmd file is written to the backup_script_location directory.

- 4 Run the **cold_backup.cmd** file.

Example:

```
c:\coldbackupscript cold_backup.cmd
```

The database files are written to the backup_files_location directory.

- 5 When the batch file has finished running, access the backup_files_location directory and zip the database files there.

UNIX Environment

The following procedure explains how to back up an Oracle database in cold mode in a UNIX environment.

To customize and run the cold backup script:

- 1 Copy the script in “The coldbackup.sh Script for the UNIX Environment” on page 202 to a text editor. Save the file as **coldBackup.sh**.
- 2 Copy the script in “The maincoldbackup.sh Script” on page 205 to a text editor. Save the file as **maincoldbackup.sh**.
- 3 Open the **coldbackup.sh** file in a text editor and change the following lines to reflect your Oracle home directory and Oracle password:

```
ORACLE_HOME=/opt/oracle9/app/oracle/product/9.2.0.4  
ORACLE_SID=oracle02  
O_CONNECT="sys/sys as sysdba"  
O_INIT=/opt/oracle9/app/oracle/admin/oracle02/pfile.ora  
O_PWFILe=${ORACLE_HOME}/database/PWDoracle02.ORA
```


- 4** Open a command line and run the following command:

```
maincoldBackup.sh backup_script_location backup_files_location
```

where

- ▶ **backup_script_location** – the directory to which the scripts are backed up
- ▶ **backup_files_location** – the directory where the files the scripts are backed up to are located.

Example:

```
maincoldBackup.sh /opt/oracle/backupscript/ /opt/oracle/backupfiles
```

The cold_backup.sh file is written to the backup_script_location directory.

- 5** Run the **cold_backup.sh** command.

The database files are written to the backup_files_location directory.

- 6** When the batch file has finished running, access the backup_files_location directory and zip the database files there.

Backup Scripts

Use the following scripts to back up an Oracle database in cold mode.

This section contains the following topics:

- ▶ “The coldbackup.cmd Script for Windows Environment” on page 198
- ▶ “The coldbackup.sh Script for the UNIX Environment” on page 202
- ▶ “The maincoldbackup.sh Script” on page 205

The coldbackup.cmd Script for Windows Environment

Use the following script to back up the Oracle database in cold mode in a Windows environment:

```
@echo off
rem This script will create the scripts necessary for a complete cold
rem backup of an Oracle database on NT.
rem Datafiles, controlfiles, redo log files and instance parameter file
rem are backed up in this script.
rem These scripts can then be run in batch. Use the AT scheduler to
rem schedule the backup job.
rem
rem Edit the SID, CONNECT and INIT strings used in this command file.
rem
REM echo COLD_GEN.CMD Usage:
REM echo Enter COLD_GEN SCRIPT_TARGET BACKUP_TARGET
REM echo SCRIPT_TARGET: is the location for the backup scripts
REM echo          e.g. F:\Backup\Scripts
REM echo BACKUP_TARGET: is the location for the Oracle file
REM echo          backups when batch is executed
REM echo          e.g. F:\Backup\Repdb2
REM echo.
REM pause
REM
REM Examples to schedule the generation and execution of command files:
REM C:\> cold_gen.cmd C:\Backup\Scripts C:\Backup\Data >
C:\Backup\Scripts\cold_gen.log
REM at 06:45pm /every:M,T,W,Th,F cmd /c
"F:\Backup\Scripts\cold_backup.cmd >
REM F:\Backup\Scripts\cold_backup.log"
REM
REM The commands above redirect the output to log files (cold_gen.log and
REM cold_backup.log) which allow you to check the successfull /unsuccessfull
REM execution.
REM example uses SID=ORCL92
REM          %ORACLE_HOME%=C:\Oracle\Ora92

REM !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```

REM These values cannot be derived, please set them to reflect your
environment
REM !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
set ORACLE_HOME=C:\Oracle\ora92
set ORACLE_SID=ORCL92
set O_CONNECT="sys/sys as sysdba"
set O_INIT=C:\Oracle\ADMIN\ORCL92\pfile\init.ora
set O_PWFIL=%%ORACLE_HOME%\database\PWDORCL92.ORA

rem Oracle Binaries
set O_PLUS=%%ORACLE_HOME%\Bin\sqlplus.exe

set O_BACKPATH=%%2
set O_SCRIPTPATH=%%1
if %%O_SCRIPTPATH%.==. goto help
if %%O_BACKPATH%.==. goto help
rem *****
rem COLD BACKUP OF THE DATABASE
rem *****

echo.
echo *****
echo -- Generate cold_backup.cmd command file to coordinate all
echo cold backup activities - for AT scheduling
echo *****
echo.
echo date /t >%%O_SCRIPTPATH%\cold_backup.cmd
echo time /t >>%%O_SCRIPTPATH%\cold_backup.cmd
ECHO REM Shutdown the database for backup purposes
>>%%O_SCRIPTPATH%\cold_backup.cmd
echo %%O_PLUS% %%O_CONNECT% @%%O_SCRIPTPATH%\shutdown.sql
>>%%O_SCRIPTPATH%\cold_backup.cmd
ECHO REM Backup the init file >>%%O_SCRIPTPATH%\cold_backup.cmd
echo copy %%O_INIT% %%O_BACKPATH%
>>%%O_SCRIPTPATH%\cold_backup.cmd
ECHO REM Backup the password file >>%%O_SCRIPTPATH%\cold_backup.cmd
echo copy %%O_PWFIL% %%O_BACKPATH%
>>%%O_SCRIPTPATH%\cold_backup.cmd

```

```

ECHO REM Run the copy_backup.cmd command file to copy all database files
>>%O_SCRIPTPATH%\cold_backup.cmd
echo start /wait %O_SCRIPTPATH%\copy_backup.cmd
>>%O_SCRIPTPATH%\cold_backup.cmd
ECHO REM Startup the database after backup has completed
>>%O_SCRIPTPATH%\cold_backup.cmd
echo %O_PLUS% %O_CONNECT% @%O_SCRIPTPATH%\startup.sql
>>%O_SCRIPTPATH%\cold_backup.cmd
echo date /t >>%O_SCRIPTPATH%\cold_backup.cmd
echo time /t >>%O_SCRIPTPATH%\cold_backup.cmd

echo.
echo *****
echo -- Create the shutdown.sql script to shutdown the database
echo  prior to backup
echo *****
echo.
echo shutdown immediate >>%O_SCRIPTPATH%\shutdown.sql
echo exit >>%O_SCRIPTPATH%\shutdown.sql
echo.
echo *****
echo -- Create the startup.sql script to startup the database
echo  after backup
echo *****
echo.
echo startup pfile=%O_INIT% >>%O_SCRIPTPATH%\startup.sql
echo exit >>%O_SCRIPTPATH%\startup.sql
echo.
echo *****
ECHO -- Create the sqlplus.sql script to obtain the names
echo  off all database files and to make an additional 'logical'
echo  backup of the trace file
echo *****
echo.
echo set heading off; >%O_SCRIPTPATH%\sqlplus.sql
echo set feedback off; >>%O_SCRIPTPATH%\sqlplus.sql
echo set linesize 1000; >>%O_SCRIPTPATH%\sqlplus.sql
echo spool %O_SCRIPTPATH%\copy_backup.cmd;
>>%O_SCRIPTPATH%\sqlplus.sql

```

```

echo select 'copy ' ^|^| name ^|^| ' %O_BACKPATH%' from v$datafile;
>>%O_SCRIPTPATH%\sqlplus.sql
echo select 'copy ' ^|^| name ^|^| ' %O_BACKPATH%' from v$controlfile;
>>%O_SCRIPTPATH%\sqlplus.sql
echo select 'copy ' ^|^| member ^|^| ' %O_BACKPATH%' from v$logfile;
>>%O_SCRIPTPATH%\sqlplus.sql
echo select 'exit' from dual; >>%O_SCRIPTPATH%\sqlplus.sql
echo spool off; >>%O_SCRIPTPATH%\sqlplus.sql
echo alter system set user_dump_dest='%O_BACKPATH%';
>>%O_SCRIPTPATH%\sqlplus.sql
echo alter database backup controlfile to trace;
>>%O_SCRIPTPATH%\sqlplus.sql
echo exit; >>%O_SCRIPTPATH%\sqlplus.sql
echo.
echo *****
ECHO -- Run the SQL*Plus script to create the copy_backup.cmd
echo *****
echo.
%O_PLUS% %O_CONNECT% @%O_SCRIPTPATH%\sqlplus.sql

echo.
echo *****
ECHO -- Generate cold_backup.cmd command file complete
echo *****

echo.
goto END_OF_FILE;

rem *****
rem USER HELP
rem *****
:HELP
echo.
echo COLD_GEN.CMD Usage:
echo Enter COLD_GEN SCRIPT_TARGET BACKUP_TARGET
echo SCRIPT_TARGET: is the location for the backup scripts
echo           e.g. F:\Backup\Scripts
echo BACKUP_TARGET: is the location for the Oracle file
echo           backups when batch is executed

```

```

echo          e.g. F:\Backup\Repdb2
goto END_OF_FILE

:HELP2
echo.
echo Error - Cannot write to %O_BACKPATH%
echo.
goto END_OF_FILErem
*****
rem HANDLE ERRORS HERE
rem *****
findstr /in "error" %O_BACKPATH%\backup.log && findstr /in "error"
%O_BACKPATH%\backup.log >%O_BACKPATH%\error.log
findstr /in "ora-" %O_BACKPATH%\backup.log && findstr /in "ora-"
%O_BACKPATH%\backup.log >%O_BACKPATH%\error.log
findstr /in "cannot" %O_BACKPATH%\backup.log && findstr /in "cannot"
%O_BACKPATH%\backup.log >%O_BACKPATH%\error.log
findstr /in "not logged" %O_BACKPATH%\backup.log && findstr /in "not logged"
%O_BACKPATH%\backup.log >%O_BACKPATH%\error.log
findstr /in "failure" %O_BACKPATH%\backup.log && findstr /in "failure"
%O_BACKPATH%\backup.log >%O_BACKPATH%\error.log
if exist %O_BACKPATH%\error.log c:pause

endlocal
:END_OF_FILE

```

The coldbackup.sh Script for the UNIX Environment

Use the following script to back up the Oracle database in cold mode in a UNIX environment:

```

#!/bin/sh

#ORACLE_HOME=/opt/oracle9/oracle/product/9.2.0.4
ORACLE_SID=shoulder
O_CONNECT="sys/sys as sysdba"

# Oracle Binaries
O_PLUS=sqlplus

```

```

O_BACKPATH=$2
O_SCRIPTPATH=$1

if [ ! -d "${O_BACKPATH}" ] ; then
echo No such directory ${O_BACKPATH}
exit `bin/false`
fi
if [ ! -d "${O_SCRIPTPATH}" ] ; then
echo No such directory ${O_SCRIPTPATH}
exit `bin/false`
fi

echo
echo "*****"
echo -- Generate cold_backup.cmd command file to coordinate all
echo cold backup activities - for AT scheduling
echo "*****"
echo
cat > ${O_SCRIPTPATH}/cold_backup.cmd << EOF
date
${O_PLUS} "${O_CONNECT}" @${O_SCRIPTPATH}/shutdown.sql
${O_SCRIPTPATH}/copy_backup.sh
${O_PLUS} "${O_CONNECT}" @${O_SCRIPTPATH}/startup.sql
date
EOF
chmod u+x ${O_SCRIPTPATH}/cold_backup.cmd
echo
echo "*****"
echo -- Create the shutdown.sql script to shutdown the database
echo prior to backup
echo "*****"
echo
cat > ${O_SCRIPTPATH}/shutdown.sql << EOF
shutdown immediate
exit
EOF

echo
echo "*****"

```

```

echo -- Create the startup.sql script to startup the database
echo  after backup
echo  "*****"
echo
cat > ${O_SCRIPTPATH}/startup.sql << EOF

```

```

startup
exit

```

EOF

```

echo
echo  "*****"
echo -- Create the sqlplus.sql script to obtain the names
echo  off all database files and to make an additional 'logical'
echo  backup of the trace file
echo  "*****"
echo

```

```

cat > ${O_SCRIPTPATH}/sqlplus.sql << EOF
set heading off;
set feedback off;
set linesize 1000;
spool ${O_SCRIPTPATH}/copy_backup.sh;
select 'cp ' || name || ' ${O_BACKPATH}' from v$datafile;
select 'cp ' || name || ' ${O_BACKPATH}' from v$controlfile;
select 'cp ' || member || ' ${O_BACKPATH}' from v$logfile;
select 'exit' from dual;
spool off;
alter system set user_dump_dest='${O_BACKPATH}';
alter database backup controlfile to trace;
exit;

```

EOF

```

chmod u+x ${O_SCRIPTPATH}/copy_backup.sh
echo
echo  "*****"
echo -- Run the SQL*Plus script to create the copy_backup.cmd
echo  "*****"

```



```

echo
${O_PLUS} "${O_CONNECT}" @${O_SCRIPTPATH}/sqlplus.sql
echo
echo "*****"
echo -- Generate cold_backup.cmd command file complete
echo "*****"
exit `bin/true`

```

The maincoldbackup.sh Script

Use the following script in the procedure to back up the Oracle database in cold mode in a UNIX environment:

```

#!/bin/sh
O_BACKPATH=$2
O_SCRIPTPATH=$1

if [ ! -d "${O_BACKPATH}" ] ; then
    echo No such directory ${O_BACKPATH}
    exit `bin/false`
fi
if [ ! -d "${O_SCRIPTPATH}" ] ; then
    echo No such directory ${O_SCRIPTPATH}
    exit `bin/false`
fi
./coldbackup.sh ${O_SCRIPTPATH} ${O_BACKPATH}
2>>${O_SCRIPTPATH}/backup.log
exit `bin/true`

```


A

Discovery Methods

Mercury Application Mapping discovery methods hold the logic for the discovery of IT infrastructure components from layers 2 through 7. This chapter describes the list of all supported discovery methods, the protocol type that is being used to communicate with the device/application, the data being discovered and the required permissions/ prerequisites for each type of protocol and operating system.

Discovered Domain	Via Protocol	Prerequisites/ Permissions	Discovered data	Notes
Layer 2	SNMP	SNMP community string with "GET" permission.	<ul style="list-style-type: none">▶ concentrator (switch)▶ port	
IP sweep	ICMP	N/A	IP	
Network infrastructure	SNMP	SNMP community string with "GET" permission.	<ul style="list-style-type: none">▶ host▶ IP▶ interface▶ SNMP▶ network▶ bridge▶ port▶ layertwo links▶ backbone links▶ route links▶ bridge links	

Appendix A • Discovery Methods

Discovered Domain	Via Protocol	Prerequisites/Permissions	Discovered data	Notes
	Telnet	Regular (non-root) operating system user.	<ul style="list-style-type: none"> ➤ host ➤ IP ➤ interface ➤ Telnet ➤ network 	
	SSH	Regular (non-root) operating system user	<ul style="list-style-type: none"> ➤ host ➤ IP ➤ interface ➤ Telnet ➤ network 	
	WMI	Admin user	<ul style="list-style-type: none"> ➤ host ➤ IP ➤ interface ➤ wmi ➤ network 	
	NetBIOS	Admin user	<ul style="list-style-type: none"> ➤ host ➤ IP ➤ interface ➤ ntcmd ➤ network 	
	Java	Etc/hosts	DNS name for IP.	
Server inventory	SNMP	SNMP community string with "GET" permission	<ul style="list-style-type: none"> ➤ disk ➤ printq ➤ program ➤ service ➤ software ➤ operating ➤ system user 	Server inventory can be discovered by: <ul style="list-style-type: none"> ➤ SNMP ➤ Telnet ➤ WMI ➤ NetBios ➤ BB

Discovered Domain	Via Protocol	Prerequisites/Permissions	Discovered data	Notes
	Telnet	Regular (non-root) operating system user	<ul style="list-style-type: none"> ➤ disk ➤ daemon ➤ software ➤ program. 	
	NetBIOS	Admin User	<ul style="list-style-type: none"> ➤ service ➤ software ➤ disk ➤ program 	
	WMI	Admin User	<ul style="list-style-type: none"> ➤ CPU ➤ disk ➤ memory ➤ program ➤ service 	
	Big Brother	N/A	<ul style="list-style-type: none"> ➤ CPU ➤ memory ➤ program ➤ disk ➤ nt ➤ eventlog ➤ program ➤ service 	Monitors the BB agent.
TCP connections	SNMP	SNMP community string with "GET" permission	<ul style="list-style-type: none"> ➤ IPserver ➤ IPclient ➤ TCP links 	
	NetBIOS	Administrator Windows User/Password	<ul style="list-style-type: none"> ➤ IPserver, ➤ IPclient ➤ TCP links 	
	Telnet	Local Admin User	<ul style="list-style-type: none"> ➤ IPserver ➤ IPclient ➤ tcp links 	

Appendix A • Discovery Methods

Discovered Domain	Via Protocol	Prerequisites/Permissions	Discovered data	Notes
	SSH	Regular (non-root) operating system user	<ul style="list-style-type: none"> ▶ IPserver ▶ IPclient ▶ TCP links 	
Microsoft domains	WIN API	N/A	<ul style="list-style-type: none"> ▶ msdomain ▶ nt 	
DB2	SQL		DB2user	Data source: ivh.userfuntion* pattern is being enhanced Versions 7 and 8
Oracle	SQL	DB user with "READ" permission from V\$ and DBA_ tables	<ul style="list-style-type: none"> ▶ dbaobjects ▶ dbarchivefile ▶ dbclient ▶ dbcontrolfile ▶ dbdatafile ▶ dbextent ▶ dbindex, dbjob ▶ dblinkobj ▶ dbredofile ▶ dbsegment ▶ dbsnapshot ▶ dbtable ▶ dbtablespace ▶ dbuser ▶ owner ▶ program 	Data sources: dba_data_files, dba_db_links, dba_jobs, dba_objects, dba_snapshots, dba_tablespaces, dba_users, v\$backup, v\$controlfile, v\$database, v\$datafile, v\$log, v\$logfile, v\$parameter, v\$recover_file, v\$session Versions 8.x & 9i

Discovered Domain	Via Protocol	Prerequisites/Permissions	Discovered data	Notes
SQL Server	SQL	Access to 'master' tables	<ul style="list-style-type: none"> ➤ sqldatabase ➤ sqlbackup ➤ sqlalert ➤ sqljob ➤ sqljobstep ➤ sqlperformance ➤ monitor ➤ sqlprocesses ➤ program ➤ dbclient ➤ sqlfile ➤ disk 	Data sources: <ul style="list-style-type: none"> ➤ sysprocesses ➤ sysdatabases ➤ backupset ➤ sysalerts ➤ sysjobs ➤ sysloginsysjobhi story ➤ sysjobschedules ➤ sysperfinfo Versions 7 and 8
Sybase	SQL		sybasedb	Data source: sysdatabases Versions 11 and 12
Exchange Server	WMI	Local Admin User	<ul style="list-style-type: none"> ➤ exchangeserver, ➤ exchangesite ➤ exchangeroutinggroup ➤ exchangeconnector, ➤ exchangelink ➤ exchangequeue 	
Citrix	SNMP	SNMP community string with "GET" permission	<ul style="list-style-type: none"> ➤ citrixserver ➤ citrixfarm ➤ citrixsession ➤ citrixclient 	

Appendix A • Discovery Methods

Discovered Domain	Via Protocol	Prerequisites/ Permissions	Discovered data	Notes
WebSphere_MQ	Telnet	UNIX Operating System Account on an MQ server that allows SUDO access to : ▶ clusqmgr ▶ dspmq ▶ runmqtsr	▶ mqqueuemanager ▶ mqcluster ▶ mqxmitq ▶ mqqueueelocal ▶ mqqueueeremote ▶ mqaliasq ▶ mqqueue ▶ mqalias ▶ mqchcdr ▶ mqchsvr ▶ mqchannel ▶ mqchannelof ▶ mqchrqstr ▶ mqchclntconn ▶ mqchclusrcvr ▶ mqchclusdr ▶ webspheremq	Version 5.3
	NetBIOS	Administrator Windows User/Password		
Weblogic	JMX	JMX MBean Server User/Password	▶ Jboss ▶ jmsdestination ▶ jmsserver ▶ ejbcomponent ▶ webapplication ▶ servlet ▶ connectionpool ▶ j2ecluster	Versions 6.1, 7.0 and 8.1

Discovered Domain	Via Protocol	Prerequisites/ Permissions	Discovered data	Notes
JBoss	JMX	JMX MBean Server User/Password	<ul style="list-style-type: none"> ➤ Jboss ➤ jmsdestination ➤ jmsserver ➤ ejbcomponent ➤ webapplication ➤ servlet ➤ connectionpool ➤ j2eecluster 	Versions 2.3, 3.2, and 4
WebSphere Application Server	JMX	JMX MBean Server User/Password		Versions 5.0 and 5.1
SAP	BAPI	SAP user/password for SAPGUI	<ul style="list-style-type: none"> ➤ SAP server ➤ SAP site ➤ SAP service ➤ SAP support package ➤ SAP component 	Versions 3 and 4
Siebel	Siebel protocol	Siebel user/password for the Server Manager utility	<ul style="list-style-type: none"> ➤ Siebel appserver ➤ Siebel compgrp ➤ Siebel component ➤ Siebel gateway ➤ Siebel site ➤ Siebel wse ➤ Siebel webapp ➤ Siebel webserver 	Versions 7.5 and 7.7
	WMI	Admin User	<ul style="list-style-type: none"> ➤ database ➤ dbconnector 	

Appendix A • Discovery Methods

Discovered Domain	Via Protocol	Prerequisites/ Permissions	Discovered data	Notes
	Telnet	Regular (non-root) operating system user	<ul style="list-style-type: none"> ▶ database ▶ Siebel application server ▶ siebelwse ▶ siebelgateway ▶ siebelsite ▶ siebelwebapp ▶ webserver 	
	SSH	Regular (non-root) operating system user	<ul style="list-style-type: none"> ▶ database ▶ Siebel App server ▶ siebelwse ▶ siebelgateway ▶ siebelsite ▶ siebelwebapp ▶ webserver 	
	NetBIOS	Admin User	<ul style="list-style-type: none"> ▶ Siebelwse ▶ Siebel gateway ▶ Siebel site ▶ Siebel webapp ▶ Siebel webserver 	
FTP	FTP	Login user	<ul style="list-style-type: none"> ▶ ftp ▶ ftp files 	
LDAP	LDAP		Active directory	

Index

A

- access rights 14
 - of users and roles 10
 - removing 15
- Ack, event action 20
- ackby, event attribute 49
- acktime, event attribute 49
- ACL Properties 136
- actions
 - adding to event rules 25, 37
 - of event rules, adding 25, 37
 - of event rules, setting in order 29
 - of roles 8
 - scheduling 175
- active events 18
 - ackby 49
 - acknowledging 20
 - acktime 49
 - ACTIVEEVENT table 44
 - ActiveEventBulkSize 43
 - ActiveEventBulkTime 43
 - activetime 50
 - checkSyncActiveEvent.cmd 44
 - comment 50
 - configuring automatic saving of 43
 - counter 49
 - createcounter 49
 - data1-10 50
 - deleting from database 44
 - Event Status 50
 - in Object Events tab 20
 - initActiveEvent.cmd 42
 - isack 49
 - issuppress 50
 - label 49
 - lastsystemtime 49
 - lastusertime 49
 - manually saving 44
 - note 49
 - replacecounter 50
 - saveActiveEventsToDb.cmd 42, 44
 - saving using Event Utilities 42
 - saving, automatic 42
 - supprestime 50
 - synchronization with objects 44
- ActiveEventBulkSize 43
- ActiveEventBulkTime 43
- activetime, event attribute 50
- additional attributes
 - changing display name 39
 - defining an Event Rule action 41
 - defining values 39
- Admin Protocol 87
- Administrator 4
- all sub-folder 112
- apilog4j-local.properties 113
- apilog4j-probe.properties 112
- appilogConfig.properties 122
 - Exporting scheduler, defining 145
- appilog-remote.properties 91
 - Parameters for discovery methods 103
- Archive Log Files 187
- attributename 46
- attributes
 - ackby 49
 - acktime 49
 - activetime 50
 - attributename 46
 - attributevalue 46
 - classname 46
 - counter 49
 - createcounter 49
 - createtime 46
 - data1-6 50

Index

- Event Status 50
- eventbase_attributename 46
- eventbase_attributevalue 46
- eventbase_category 48
- eventbase_classname 46
- eventbase_eventid 46
- eventbase_message 46
- eventbase_severity 46
- eventbase_usertime 46
- eventid 46
- ID 48
- isack 49
- issuppress 50
- label 49
- lastsystemtime 49
- lastusertime 49
- message 46
- note 49
- param 47
- rawevent_param 47
- rawevent_subsystem 47
- rawevent_system 47
- replacecounter 50
- root_container 48
- root_createtime 46
- severity 46
- subsystem 47
- supprestime 50
- system 47
- usertime 46
- attributevalue 46
- B**
- backup methods 192
- backup scripts 197
- backup sub-directory 145
- C**
- checkSyncActiveEvent.cmd 44
- classes
 - in time rules 36
 - mapping objects to 106
- classname 46
- cold backup script
 - customizing and running 195
- cold_export.cmd 145
- Collectors 123
- collectors directory 110
- comment, event attribute 50
- configuration parameters
 - configuring 92
- configuration saving and loading, user-defined 147
- connection data
 - defining 60
- connection data for protocol
 - configuring 59
- connection protocols
 - ports 93
- Contract tab
 - description 69
- counter, event attribute 49
- createcounter, event attribute 49
- createtime 46
- customized packages 160
 - creating 161
 - uninstalling and updating 164
 - validating 162
 - verifying validity 162
- D**
- data1-6, event attribute 50
- database
 - building 144
- database load behavior 189
- databases
 - building 143, 144
 - folder 97, 98
 - importing 146
 - manually exporting 144
 - OnlineDBCreator.cmd 143
- default
 - users 4
- delCollectors.bat 111
- descriptor.xml 161
- design view tab 68
- directories 137
- Discovery Management 51
- discovery methods 207

- administrative information of 100
- configuration files 103
- configuring 103
- performing administrative actions in
 - UNIX 81
- threads allocated to 101
- discovery patterns
 - activating 76
 - defining 75
 - designing 70
 - editing 74
 - editing in Source View tab 74
 - exporting to file 66
 - saving to file 76
- discovery process
 - configuration files 103
 - configuration management 90
 - configuration parameters 91
 - configuring 89
 - defining according to MS domain
 - types 104
 - defining according to server Windows
 - types 105
 - managing 52
- discovery scope
 - configuring 56
- discovery system
 - directory location, defining 123
 - directory structure 110
- discovery task data from Probe Gateway
 - deleting 111
- discovery task data from Probe Manager
 - deleting 111
- discovery tasks
 - data in Probe Gateway, Probe
 - Manager, deleting and rebuilding 111
 - in Probe Manager 114
 - limited 123
 - requests, See discovery tasks requests 98
 - results, See discovery tasks results 98
 - schedule 114, 123
- discovery tasks requests
 - number of in one bulk 97
 - Probe Gateway waiting time for 98

- discovery tasks results
 - number of parallel connections for
 - sending 97
 - saving, number of retries 123
 - stored in Probe Gateway repository 98
- discoveryManager folder 111
- documentation
 - typographical conventions x
- Domain scope
 - configuring 56
- domains
 - Microsoft types, defining the
 - discovery process according to 104
- dynamic management states
 - creating 177
- dynamic objects
 - states 177

E

- Event Rule
 - configuring 23
 - defining 24
 - deleting 26
 - editing 25
- Event Rule actions
 - defining 29
 - description 30
 - removing 30
- Event Rule tab
 - understanding 23
- event rules
 - defining 23
 - order 25
 - ordering 25
- event status, event attribute 50
- Event System 123
 - managing 17
 - workflow 22
- Event System architecture 21
- eventbase_attributename 46
- eventbase_attributevalue 46
- eventbase_category 48
- eventbase_classname 46
- eventbase_eventid 46
- eventbase_message 46

Index

- eventbase_severity 46
- eventbase_usertime 46
- eventid 46
- events
 - log 18
 - managing 20
 - raw and active 18
 - selecting for display 2
- events.log 44
- Export Database Properties 135

- F**
- FTP
 - protocol definitions 82
- ftpprotocol 82

- G**
- general configuration 95
- General Stuff Properties 134
- groups
 - actions of 8
- Guest 4
- Gui Resource Path 136

- H**
- hot_export.cmd 42
 - exporting 42
- HTML adapter 94
- HTTP protocol
 - definition in appilog-remote.properties 92
 - HTML adapter 94
 - number of parallel connections for sending task results 97
 - number of task requests in one bulk 97
 - servlet definitions 93

- I**
- IBM HTTP Server
 - protocol definitions 85
- ibmhttpserverprotocol 85
- icons
 - adding customized icons 117
 - creating customized body icon 118
 - creating customized family icon 119
 - customized 117
 - name format 117
- ID, event attribute 48
- import.cmd 146
- initActiveEvent.cmd 42, 44
- interfaceType.xml 104
- IP address range
 - rules 58
 - rules for defining 58
- isack, event attribute 49
- issuppress, event attribute 50

- J**
- JBOSS
 - protocol definitions 80
- jbossprotocol 80
- JMS subsystem location 123
- jms.properties 111

- L**
- label of objects, size of 131
- label, event attribute 49
- lastsystemtime, event attribute 49
- lastusertime, event attribute 49
- links
 - inserting 151
- localResults folder 100
- log files 137
- log load
 - reducing 141
- log properties
 - defining 139
- logs
 - events 18
 - output, controlling 121
 - pql_statistics.log 127
 - pqlfuse.log 124

- M**
- Map Server 129
- Mercury Application Mapping server

- shutting down 136
- starting 136
- message
 - event attribute 46
- Microsoft domains, defining discovery
 - process according to 104
- modules
 - configuring 63
 - creating 64
 - editing definition 65
 - exporting patterns in module to file 66
 - saving definition to file 65
- Modules Tab
 - understanding 53
- Modules tab 53
- modules tab 53
- Monitoring CPU 186
- msDomainNames.xml 104
- msServerTypes.xml 105

N

- naming conventions
 - XML files 160
- network interface type, converting 104
- new domain
 - defining 56
- note, event attribute 49
- NT Admin
 - protocol definitions 87
- ntadminprotocol 87

O

- Object Events tab 20
- objects
 - inserting 149
 - labels, number of characters 131
 - mapping to classes 106
 - synchronization
 - with Active Events 44
- oidToHostClass.xml 106
- OnlineDBCcreator.cmd 143
- Oracle
 - backup guidelines 191

- configuring, monitoring and tuning 181
- monitoring and tuning guidelines 186
- protocol definitions 85
- Recovery Manager 194
- Oracle Alert File 186
- Oracle configuration
 - guidelines 182
- oracleprotocol 85
- OsDescriptionToType.xml 107

P

- p2pViewer Folder 112
- packages
 - creating 161
 - creating customized 153, 160
 - creating customized packages 161
 - customized 160
 - dependencies 157
 - deployment 157
 - location 157
 - structure 154
 - uninstalling 164
- Packages tab 53
 - description 55
- packageVerify.cmd 162
- packaging.xml 160
- param, event attribute 47
- patches Folder 112
- Pattern Editor
 - understanding 67
- patterns
 - editing 69
 - exporting module patterns to file 66
 - saving to file 76
- patternToClass.xml 111
- performance monitoring
 - discovery pattern location 112
 - patternToClass.xml 111
- pm sub-folder 112
- portNumberToPortName.xml 107
- pql_statistics.log 127
- predefined tasks
 - description 168

Index

- probe
 - serverData folder 113
 - Probe Gateway
 - apilog4j-probe.properties 112
 - configuration file 91
 - connection with server 92
 - connections 92
 - databases folder 97, 98
 - delCollectors.bat 111
 - deleting and rebuilding discovery task data in 111
 - domain 93
 - general definitions 97
 - log, configuration file of 112
 - number of open connections to main repository 97
 - receiving task requests 97, 98
 - sending task results 97, 98
 - serverData folder 113
 - time intervals for procedures 98
 - Probe Manager
 - apilog4j-local.properties 113
 - configuration file 91
 - delCollectors.bat 111
 - deleting and rebuilding discovery task data 111
 - domain 93
 - general definitions 101
 - log, configuration file of 113
 - performing administrative actions in UNIX 87
 - time intervals for procedures 98
 - probeGateway Folder 112
 - probeManager Folder 113
 - profile, of user 2
 - protocols
 - admin 87
 - configuring connection data for 59
 - definitions 77
 - deleting 62
 - deleting connection details 62
 - editing connection details 63
 - FTP 82
 - pstools 87
 - Telnet 81
 - WebLogic 80, 81
 - WMI 79, 80, 81
 - XCMD 87
 - Proxy 132
 - pstools 87
 - Pstools protocol 87
- ## R
- RAID 184
 - Raw and Active Events
 - attributes 45
 - raw events 18
 - attributename 46
 - attributevalue 46
 - classname 46
 - createtime 46
 - eventbase_attributename 46
 - eventbase_attributevalue 46
 - eventbase_category 48
 - eventbase_classname 46
 - eventbase_eventid 46
 - eventbase_message 46
 - eventbase_usertime 46
 - eventid 46
 - ID 48
 - log 18
 - message 46
 - param 47
 - rawevent_param 47
 - rawevent_subsystem 47
 - rawevent_system 47
 - root_container 48
 - root_createtime 46
 - severity 46
 - subsystem 47
 - system 47
 - usertime 46
 - rawevent_parameter 47
 - rawevent_subsystem 47
 - rawevent_system 47
 - remoteAgent Folder 114
 - replacecounter, event attribute 50
 - Report System 132
 - resources
 - packages 156
 - RMAN 194

- RMI protocol
 - definition in appilog-remote.properties 92
- roles 4
 - access rights 10
 - creating 8
 - deleting 10
 - editing 9
 - management 4
- root_container 48
- root_createtime 46
- Rule Condition
 - defining 26
 - editing 28
 - removing 28

- S**
- SAP
 - protocol definitions 84
- sapprotocol 84
- saveActiveEventsToDb.cmd 42, 44
- SaveConfiguration.cmd 147
- schedule
 - database export 145
 - discovery tasks 114, 123
- scheduled actions 169
- Scheduler
 - exporting 145
- Security Manager
 - understanding 11
- server
 - connection with the Probe Gateway 92
 - pause length 133
 - saving task results, number of retries 123
 - server_shutdown.cmd 42
 - subsystem parameters, configuring 122
 - Windows types, defining discovery process according to 105
- Server connections 92
- server login data
 - defining 141
- server properties configuration 121
- server subsystem parameters
 - configuring 122
- server_shutdown.cmd 42
- serverData folder 98
 - interfaceType.xml 104
 - msServerTypes.xml 105
 - syslogFacility.xml 108
 - syslogSeverity.xml 109
 - tcpLinkTypesList.xml 109
- serverdata folder
 - msDomainNames.xml 104
 - OsDescriptionToType.xml 107
- severity
 - of raw events 46
- SGA 185
- Siebel Gateway
 - protocol definitions 83
- siebelgtwyprotocol 83
- SNMP
 - protocol definitions 78
- snmpprotocol 78
- Source View tab
 - description 68
- SQL
 - protocol definitions 78
- sqlpprotocol 78
- sshprotocol 82
- states
 - using 179
- statistics
 - collecting 189
- subsystem
 - event attribute 47
- Suppress, event action 20
- supprestime, event attribute 50
- synchronization
 - Objects and Active Events 44
- SysLog 108, 109
- syslogFacility.xml 108
- syslogSeverity.xml 109
- System Global Area (SGA) 185
- system, event attribute 47

- T**
- Tablespace Storage Space 187

Index

tasks

- activating 170
- creating 169
- predefined 168
- scheduling 167, 170

TCP conversion list 109

tcpLinkTypesList.xml 109

Telnet

- discovering operating systems through 107
- protocol definition 81
- protocol definitions 81

telnetprotocol 81

thread pools 95

time intervals

- for time rules 36
- of requests for updates from the server 116

Time Rule 22

- activating 36
- active 36
- adding 36, 38
- class 36
- configuring 34
- defining 34, 36, 38
- deleting 38
- editing 38
- time interval 36

time rules

- adding 38

Time Rules tab

- understanding 35

topology databases

- adding objects 148
- building 143
- exporting 144
- exporting automatically 145
- importing 146
- management 143

TQL calculations

- saving 127

TQL circuit breakers 124

TQLs 124

- rebuilding 127
- saving calculations 127
- user-defined, saving and loading 147

typographical conventions x

U

user

- creating a new user 6

user interface

- properties 116
- settings 115
- time interval between each request for updates from the server 116

user management 4

user profiles

- defining 2
- enabling administrator to define 7

user properties

- modifying 7
- viewing and modifying 7, 8

users

- access rights 10
- deleting 8

usertime

- event attribute 46

utils folder 147

V

validation

- verifying for customized package 162

vendor

- in raw and active events 48

view settings

- customizing 2

View tab

- designing 68

views

- selecting for display 2
- sharing 2

W

WebLogic

- protocol definitions 80, 81

weblogicprotocol 80

WebSphere

- protocol definitions 86

websphere 86

- Windows servers, defining discovery process
 - according to 105
- WMI 79, 80, 81
 - protocol definitions 79
- wmiprotocol 79
- WorldManager System 133

X

- xcmd 87
- XCMD protocol 87
- XML files
 - naming conventions 160